UNIVERSITAT POLITÉCNICA DE CATALUNYA
DOCTORADO EN INGENIERIA TELEMÁTICA

"CONTRIBUCIÓN A LA VALIDACIÓN DE CERTIFICADOS EN ARQUITECTURAS DE AUTENTICACIÓN Y AUTORIZACIÓN"

TESIS DOCTORAL

Autor: Isabel Cristina Satizábal Echavarría
Director: Jordi Forné Muñoz

Marzo 2007

Resumen

Las arquitecturas de autenticación y autorización basadas en certificados no han sido ampliamente aceptadas debido a su costo, inflexibilidad y difícil manejo.

Uno de los procesos que incrementa la complejidad de la infraestructura de clave pública (PKI) es sin duda la validación de caminos de certificación que involucra: descubrir el camino, recuperar los certificados, verificar su firma digital y constatar que ninguno haya expirado o haya sido revocado. Este proceso le exige al verificador ciertas capacidades de cálculo y almacenamiento, que pueden exceder las características de algunos dispositivos, como los teléfonos móviles y las tarjetas inteligentes.

En esta tesis evaluamos el coste computacional y la capacidad de almacenamiento requeridas por un verificador para llevar a cabo el proceso de validación de cadenas de certificados y determinamos que estos factores son críticos para los dispositivos con capacidades limitadas. Además, se presentan dos propuestas que contribuyen a simplificar el proceso de validación de caminos desde el punto de vista del verificador: TRUTHC y PROSEARCH.

TRUTHC utiliza cadenas de hash para establecer una relación de confianza alternativa entre las diferentes entidades de una PKI jerárquica. De esta manera, las operaciones de verificación de firma son sustituidas por simples operaciones de hash, lo que contribuye a disminuir el coste computacional del verificador. La validación de los caminos es realizada por una tercera parte de confianza llamada autoridad de verificación (VA).

TRUTHC es compatible con los certificados X.509 y su seguridad depende en gran medida de la confidencialidad de las semillas. Este mecanismo puede usarse en entornos donde los dispositivos no cuentan con la suficiente capacidad de procesamiento para realizar la validación y deben delegar este proceso a otra entidad, como en el caso de las redes móviles que utilizan servidores de validación.

Por otra parte, PROSEARCH establece una jerarquía virtual entre las entidades de una PKI en malla, basado en el nivel de confiabilidad de las autoridades participantes. La jerarquía

establecida facilita el proceso de construcción de caminos, pues cuando las relaciones de confianza son unidireccionales existe un único camino entre cada par de entidades.

PROSEARCH no establece nuevas relaciones de confianza entre las entidades sino que utiliza las relaciones existentes para establecer la jerarquía. Por tanto, no es necesario expedir nuevos certificados o ajustar los puntos de confianza de las entidades.

Adicionalmente, PROSEARCH se puede adaptar a entidades con capacidades limitadas de procesamiento y almacenamiento ya que la jerarquía se establece considerando una longitud máxima para los caminos de certificación.

La rápida ejecución de PROSEARCH hace que pueda aplicarse en diversos entornos como escenarios críticos y redes ad-hoc.

Aunque la jerarquía encontrada no siempre es la mejor solución, los resultados de las simulaciones muestran que en la mayoría de los casos se encuentra una jerarquía aceptable, considerando la simplicidad y fácil implementación del protocolo.

Abstract

Authentication and authorisation architectures based on certificates have not been widely accepted due to their cost, inflexibility and difficult management.

The complexity of the Public Key Infrastructure (PKI) is increased by the certification path validation process that involves: discovering the path, retrieving the certificates, verifying their digital signature and checking that none of the certificates have expired or have been revoked. This process demands certain processing and storage capacity from the verifier that can exceed the features of some devices, such as mobile phones and smart cards.

In this thesis, we evaluate the computational cost and the storage capacity required by a verifier to carry out the path validation process and determine that they are critical factors for constrained devices. In addition, we introduce two proposals that contribute to simplify the path validation process from the verifier's point of view: TRUTHC and PROSEARCH.

TRUTHC uses two hash chains to establish an alternative trust relationship among the different entities of a hierarchical PKI. Thus, the signature verification operations are replaced by hash operations, what contributes to decrease the computational cost of the verifier. The path verification is carried out by a Verification Authority (VA). TRUTHC is compatible with the X.509 certificates and its security depends on a large extent of the seeds' confidentiality. TRUTHC can be used in environments where devices have limited processing capacity and it is necessary to delegate the validation process in other entity, such as mobile networks with validation servers.

On the other hand, PROSEARCH establishes a virtual hierarchy in a mesh PKI, based on the trustworthiness level of the participant entities. This protocol facilitates the certification path discovery since in a hierarchical model the trust relationships are unidirectional and there is a single path between each pair of entities.

PROSEARCH does not establish new trust relationships among the entities but it takes the existing relationships to establish the hierarchy. Thus, it is not necessary to issue new certificates or adjust the trust points.

In addition, PROSEARCH is adaptable to entities with limited processing and storage capacities, since hierarchy is established considering a maximum certification path length.

The fast execution of PROSEARCH makes possible its use in different environments such as critical scenarios and ad-hoc networks.

Although the hierarchy found by our protocol is not always the best solution, in our opinion this is not an important drawback since simulation results show that in most cases an acceptable hierarchy is found, especially considering that the simplicity of the protocol makes it easy-to-implement even for constrained devices.

Agradecimientos

En primer lugar quiero agradecer a Dios, por darme la oportunidad de venir a un país tan lejano del mío a realizar mis estudios de doctorado. Ésta ha sido una experiencia muy gratificante pues considero que no sólo me ha enriquecido intelectualmente sino también personal y espiritualmente, gracias a la diversidad de personas que he conocido y que han compartido conmigo sus diferentes puntos de vista. Dios también me ha dado la fortaleza para seguir adelante a pesar de las dificultades, por lo que le estoy profundamente agradecida.

También quiero agradecer a mi familia, a quién le debo todo lo que soy y me ha apoyado incondicionalmente todo este tiempo a pesar de la distancia que nos separa. Espero que Dios me de la oportunidad de estar junto a ellos muy pronto.

Otra persona a la que quisiera agradecer es a mi novio Rafael, por permanecer a mi lado en esta recta final del doctorado, apoyándome, animándome y aconsejándome.

No puedo dejar de agradecer a mis compañeros del despacho "patera", que con el tiempo se han convertido en mi familia. Mil gracias a Mónica, o debería decir Dra. Mónica?, a Oscar, Patricia, John, Rafael, Pablo, Guillermo, Luis, Aída, David, por su amistad, su alegría y sabios consejos.

Agradezco a mis amigos, muy especialmente a María Isabel y Leydi, por acogerme desde mi llegada a Barcelona, escucharme y hacerme reír con sus locuras.

Muchísimas gracias a mi tutor Jordi Forné, por la gran persona que es, porque me ha sabido orientar en el inmenso mundo de la seguridad y ha invertido mucho tiempo y esfuerzo en este trabajo de investigación. Sin tí Jordi, esta meta que estoy a punto de alcanzar no hubiera sido posible, muchas gracias.

Finalmente, quiero agradecer a las personas que de una u otra manera han contribuido con su granito de arena en la realización de este trabajo y a hacer mi estancia en este país tan agradable.

Índice General

1 Introducción 1

1.1 Acerca de la Tesis 1

1.2 Motivación 2

1.3 Objetivos y Contribuciones de esta Tesis 4

1.4 Justificación 6

1.5 Organización de la Tesis 8

1.6 Publicaciones 9

1.6.1 Congresos 9

1.6.1.1 Nacionales 9

1.6.1.2 Internacionales 9

1.6.2 Revistas 10

1.6.2.1 Nacionales 10

1.6.2.2 Internacionales 11

1.6.3 Otras Publicaciones 11

2 Validación en Sistemas de Autenticación y Autorización 13

2.1 Introducción 13

2.2 Criptografía 14

2.3 Autenticación 17

2.3.1 Autenticación de Mensaje 17

2.3.2 Autenticación de Usuario 18

2.3.3 Arquitecturas de Autenticación 21

2.3.4 Infraestructura de Clave Pública (PKI) 23

2.3.5 Wireless Application Protocol PKI (WPKI) 26

2.3.6 Estándares PKI 28

2.3.6.1 X.509 28

2.3.6.2 PKIX 29
2.3.6.3 X.500 29
2.3.6.4 LDAP 30
2.3.6.5 S/MIME 30
2.3.6.6 IPSec 31
2.3.6.7 TLS 31
2.3.6.8 SPKI 31
2.3.6.9 OpenPGP 32
2.3.6.10 WAP 32
2.3.6.11 XML 32
2.4 Autorización 33
2.4.1 Modelos de Control de Acceso 33
2.4.2 Estándares del Servicio de Autorización 36
2.4.2.1 Certificados de Atributos X.509 36
2.4.2.2 SPKI/SDSI 41
2.4.2.3 SAML (Security Assertion Markup Language) 41
2.4.3 Gestión de Autorización en Entornos Web 42
2.4.3.1 Shibboleth 43
2.4.3.2 PAPI 43
2.4.3.3 Liberty Alliance 44
2.5 Validación de Certificados 45
2.5.1 Validación de Caminos de Certificación 45
2.5.1.1 Modelos de Confianza 46
2.5.1.2 Métodos de Construcción de Caminos 53
2.5.1.3 Recuperación de Certificados 54
2.5.1.4 Verificación de Firma de los Certificados 56
2.5.1.5 Revocación de Certificados 57
2.5.2 Validación de Caminos de Delegación 60
2.5.2.1 Algoritmos de Búsqueda de Caminos 61

2.5.3 Estándares para Validación de Certificados 63
2.6 Conclusiones 63
3 Coste Computacional de Validación de Caminos de Certificación y Delegación 65
3.1 Introducción 65
3.2 Definiciones y Suposiciones 66
3.3 Coste Computacional de Caminos de Certificación 67
3.3.1 Primer Caso: Sin Revocación 68
3.3.2 Segundo Caso: Con Revocación 68
3.3.2.1 Revocación con CRL 68
3.3.2.2 Revocación con OCSP 69
3.4 Coste Computacional de Caminos de Delegación 73
3.4.1 Primer Caso: Sin Revocación 73
3.4.2 Segundo Caso: Con Revocación 74
3.4.2.1 Revocación con CRL 74
3.4.2.2 Revocación con OCSP 75
3.5 Conclusiones 77
4 Capacidad de Almacenamiento en Validación de Caminos de Certificación 81
4.1 Introducción 81
4.2 Capacidad de Almacenamiento del Verificador 82
4.2.1 Primer Caso: Sin Revocación 82
4.2.2 Segundo Caso: Con Revocación 84
4.2.2.1 Revocación con CRL 84
4.2.2.2 Revocación con OCSP 85
4.3 Ejemplo de Aplicación 90
4.3.1 Primer Caso: Sin Revocación 92
4.3.2 Segundo Caso: Con Revocación 92
4.3.2.1 Revocación con CRL 92

4.3.2.2 Revocación con OCSP 93
4.4 Conclusiones 94
5 TRUTHC (Trust Relationship Using Two Hash Chains) 95
5.1 Introducción 95
5.2 Validación desde el Punto de Vista del Verificador 96
5.3 Cadenas de Hash 98
5.4 Escenario de Estudio 99
5.5 Descripción de TRUTHC 103
5.5.1 Expedición de Certificados 103
5.5.2 Verificación de Certificados 107
5.6 Integración de TRUTHC con los Certificados X.509 108
5.7 Análisis de Seguridad 109
5.8 Evaluación 110
5.8.1 Expedición de Certificados 111
5.8.1.1 PKI Típica 111
5.8.1.2 PKI con TRUTHC 111
5.8.2 Verificación de Certificados 113
5.8.2.1 PKI Típica 113
5.8.2.2 PKI con TRUTHC 113
5.9 TRUTHC en Validación de Arquitecturas de Autorización 115
5.10 Conclusiones 116
6 PROSEARCH (Protocol to Simplify the Certification Path Discovery Constructing a Hierarchy) 119
6.1 Introducción 119
6.2 Validación de Caminos en Arquitecturas Descentralizadas 120
6.3 Descripción de PROSEARCH 121
6.3.1 Orden de Confiabilidad entre las CAs 122
6.3.2 Creación de la Jerarquía 124
6.3.3 Vinculación de una Nueva CA a la Jerarquía 125

6.3.4 Salida de una CA de la Jerarquía ... 126
6.4 Análisis de Seguridad ... 126
6.5 Ejemplo de Aplicación ... 127
6.5.1 Orden de Confiabilidad entre las CAs ... 127
6.5.2 Creación de la Jerarquía ... 129
6.5.3 Tiempo de ejecución de PROSEARCH ... 132
6.5.4 Búsqueda de Caminos Simplificada ... 134
6.5.5 Longitud de Caminos de Certificación ... 135
6.5.6 Vinculación de una Nueva CA a la Jerarquía ... 136
6.5.7 Salida de una CA de la Jerarquía ... 138
6.5.8 Parámetros Falsos ... 139
6.6 PROSEARCH en Arquitecturas de Autorización ... 140
6.7 Conclusiones ... 140
7 Evaluación de PROSEARCH en Diferentes Entornos ... 143
7.1 Introducción ... 143
7.2 PROSEARCH en Infraestructuras Críticas ... 144
7.2.1 Evaluación ... 144
7.3 PROSEARCH en Redes Ad-Hoc ... 146
7.3.1 PKI y Redes Ad-hoc ... 147
7.3.1.1 CA Distribuida con Criptografía Umbral ... 148
7.3.1.2 PKI Auto-Organizada ... 148
7.3.2 Evaluación ... 149
7.3.2.1 Longitud Máxima de los Caminos de Certificación ... 150
7.3.2.2 Variación de la Proporción de Relación ... 150
7.3.2.3 Variación de la Longitud Máxima de los Caminos ... 158
7.4 Conclusiones ... 162
8 Conclusiones y Líneas Futuras ... 165
8.1 Conclusiones ... 165
8.2 Líneas Futuras ... 167

A Tablas de Datos 169

B Fases de PROSEARCH y Algoritmos 191

B.1 Orden de Confiabilidad entre las CAs 191

B.2 Creación de la Jerarquía 193

Bibliografía 195

Lista de Figuras

Figura 2.1: Arquitectura WPKI 28
Figura 2.2: Sintaxis de un certificado de atributos 37
Figura 2.3: Modelo general 38
Figura 2.4: Modelo de control de acceso 39
Figura 2.5: Modelo de delegación 40
Figura 2.6: Modelo de roles 40
Figura 2.7: PKI de una única CA 48
Figura 2.8: PKI jerárquica 49
Figura 2.9: PKI en malla 51
Figura 2.10: PKI con CA puente 52
Figura 2.11: PKI híbrida 52
Figura 3.1: Coste computacional sin y con revocación usando RSA 71
Figura 3.2: Coste computacional sin y con revocación usando ECDSA 72
Figura 3.3: Coste computacional con RSA (1024 bits) y ECDSA (163 bits) para PDA con procesador StrongARM a 200MHz 72
Figura 3.4: Coste computacional sin y con revocación de caminos de delegación 78
Figura 4.1: Capacidad de almacenamiento sin y con revocación usando RSA (1024 bits) 88
Figura 4.2: Capacidad de almacenamiento sin y con revocación usando ECDSA (163 bits) 88
Figura 4.3: Capacidad de almacenamiento con CRL usando RSA (1024 bits) 89
Figura 4.4: Capacidad de almacenamiento con CRL usando ECDSA (163 bits) 89
Figura 4.5: Capacidad de almacenamiento con RSA (1024 bits) y ECDSA (163 bits) 90
Figura 4.6: Arquitectura jerárquica de Verisign 91
Figura 4.7: Capacidad de almacenamiento en ejemplo de aplicación 93
Figura 5.1: Camino de certificados anidados 98
Figura 5.2: Escenario de estudio 99
Figura 5.3: Cadena de certificados 102

Figura 5.4: Encadenamiento de certificados mediante cadenas de hash..... 106
Figura 5.5: Modelo de verificación..... 107
Figura 5.6: Coste computacional de TRUTHC vs. PKI típica..... 115
Figura 6.1: PKI ejemplo..... 127
Figura 6.2: Información compartida..... 129
Figura 6.3: Autoridad 3 como CA superior de autoridad 2..... 130
Figura 6.4: Jerarquía y parámetros resultantes..... 131
Figura 6.5: Jerarquía establecida..... 132
Figura 6.6: Vinculación a la jerarquía..... 136
Figura 6.7: Parámetros de las CAs..... 136
Figura 6.8: Nueva jerarquía incorporando la autoridad 8..... 137
Figura 6.9: PKI sin autoridad 3..... 138
Figura 6.10: Nueva jerarquía sin la autoridad 3..... 139
Figura 6.11: Jerarquía con autoridad maliciosa..... 140
Figura 7.1: Número de CAs vs. número de rondas..... 145
Figura 7.2: Número de CAs vs. número de CAs raíz..... 146
Figura 7.3: Longitud máxima de los caminos con mejor jerarquía..... 151
Figura 7.4: Número de CAs vs. número promedio de raíces..... 151
Figura 7.5: Número de CAs vs. longitud promedio de los caminos..... 152
Figura 7.6: Longitud promedio de caminos con mejor jerarquía..... 153
Figura 7.7: Número de CAs vs. probabilidad de fallo..... 155
Figura 7.8: Número de CAs vs. número total de rondas..... 156
Figura 7.9: Número de CAs vs. mensajes enviados por cada CA..... 157
Figura 7.10: Relación entre el número de mensajes enviados y vecinas por cada CA..... 157
Figura 7.11: Número de CAs vs. mensajes firmados emitidos por cada CA..... 158
Figura 7.12: Número de CAs vs. número de raíces cuando varía L_{MAX}..... 159
Figura 7.13: Número de CAs vs. probabilidad de fallo cuando varía L_{MAX}..... 159
Figura 7.14: Número de CAs vs. longitud promedio de caminos cuando varía L_{MAX}..... 160
Figura 7.15: Número de CAs vs. número de rondas cuando varía L_{MAX}..... 161

Figura 7.16: Número de CAs vs. número de mensajes enviados por CA cuando varía L_{MAX} 161
Figura 7.17: Número de CAs vs. número de mensajes firmados por CA cuando varía L_{MAX} 162

Lista de Tablas

Tabla 3.1: Notación de coste computacional 67
Tabla 3.2: Tiempo de operaciones criptográficas en una PDA con procesador StrongARM a 200MHz 68
Tabla 3.3: Número de operaciones criptográficas en caminos de certificación 70
Tabla 3.4: Coste computacional de validación de caminos de certificación 70
Tabla 3.5: Tiempo de operaciones de verificación con RSA-1937 en teléfonos móviles 71
Tabla 3.6: Tiempo de operaciones criptográficas en ordenador con procesador Pentium 4 a 2,1GHz 73
Tabla 3.7: Tamaño del contenido de un certificado de atributos 74
Tabla 3.8: Tamaño del contenido de una respuesta OCSP básica 76
Tabla 3.9: Número de operaciones criptográficas en caminos de delegación 77
Tabla 3.10: Coste computacional de caminos de delegación 77
Tabla 4.1: Notación de capacidad de almacenamiento 83
Tabla 4.2: Tamaño de una solicitud OCSP 85
Tabla 4.3: Capacidad de almacenamiento en validación de caminos de certificación 87
Tabla 4.4: Capacidad de almacenamiento en ejemplo de aplicación 93
Tabla 5.1: Notación TRUTHC 100
Tabla 5.2: Tiempo de ejecución de operaciones criptográficas de CAs y VA 110
Tabla 5.3: Tiempo de ejecución de operaciones criptográficas del verificador 111
Tabla 5.4: Coste computacional del proceso de expedición de certificados 113
Tabla 6.1: Notación PROSEARCH 122
Tabla 6.2: Parámetros del sistema DSSS usados en el estándar 802.11b 133
Tabla 6.3: Número de caminos exitosos vs. número de caminos posibles 134
Tabla 6.4: Longitud de caminos más cortos en PKI ejemplo 135
Tabla 6.5: Longitud de caminos con PROSEARCH 135

Tabla A.1: Coste computacional sin y con revocación usando RSA 169
Tabla A.2: Coste computacional sin y con revocación usando ECDSA 170
Tabla A.3: Coste computacional con RSA (1024 bits) y ECDSA (163 bits) para PDA con procesador StrongARM a 200MHz 170
Tabla A.4: Coste computacional sin y con revocación de caminos de delegación 170
Tabla A.5: Capacidad de almacenamiento sin y con revocación usando RSA (1024 bits) .. 171
Tabla A.6: Capacidad de almacenamiento sin y con revocación usando ECDSA (163 bits) 171
Tabla A.7: Capacidad de almacenamiento con CRL usando RSA (1024 bits) 172
Tabla A.8: Capacidad de almacenamiento con CRL usando ECDSA (163 bits) 172
Tabla A.9: Capacidad de almacenamiento con RSA (1024 bits) y ECDSA (163 bits) 172
Tabla A.10: Capacidad de almacenamiento en ejemplo de aplicación 173
Tabla A.11: Coste computacional de TRUTHC vs. PKI típica 173
Tabla A.12: Número de CAs vs. número de rondas 174
Tabla A.13: Número de CAs vs. número de CAs raíz 175
Tabla A.14: Longitud máxima de los caminos con mejor jerarquía 176
Tabla A.15: Número de CAs vs. número promedio de raíces 177
Tabla A.16: Número de CAs vs. longitud promedio de los caminos 178
Tabla A.17: Longitud promedio de caminos con mejor jerarquía 179
Tabla A.18: Número de CAs vs. probabilidad de fallo 180
Tabla A.19: Número de CAs vs. número total de rondas 181
Tabla A.20: Número de CAs vs. mensajes enviados por cada CA 182
Tabla A.21: Relación entre el número de mensajes enviados y vecinas por cada CA 183
Tabla A.22: Número de CAs vs. mensajes firmados emitidos por cada CA 184
Tabla A.23: Número de CAs vs. número de raíces cuando varía L_{MAX} 185
Tabla A.24: Número de CAs vs. probabilidad de fallo cuando varía L_{MAX} 186
Tabla A.25: Número de CAs vs. longitud promedio de caminos cuando varía L_{MAX} 187
Tabla A.26: Número de CAs vs. número de rondas cuando varía L_{MAX} 188
Tabla A.27: Número de CAs vs. número de mensajes enviados por CA cuando varía L_{MAX} 189
Tabla A.28: Número de CAs vs. número de mensajes firmados por CA cuando varía L_{MAX} 190

Tabla B.1: Notación algoritmos..... 191

Lista de Acrónimos

AA: Attribute Authority.

ABAC: Attribute Based Access Control.

AC: Attribute Certificate.

ACL: Access Control List.

ACRL: Attribute Certificate Revocation List.

AD: Authenticated Dictionary.

AES: Advanced Encryption Standard.

API: Application Programming Interface.

AQM: Attribute Query Message.

ARM: Attribute Response Message.

AS: Authentication Server.

BCA: Bridge Certification Authority.

CA: Certification Authority.

CCPD: Codeword-based Certificate Path Discovery.

CP: Certificate Policy.

CPS: Certification Practice Statement.

CRL: Certificate Revocation List.

CRS: Certificate Revocation System.

CRT: Certificate Revocation Tree.

CTS: Clear To Send

CVT: Certificate Verification Tree.

DAC: Discretionary Access Control.

DAP: Directory Access Protocol.

DCE: Distributed Computing Environment.

DES: Digital Encryption Standard.

DFS: Distributed File System.

DH: Diffie Hellman.

DN: Distinguished Name.

DNS: Domain Name System.

DoS: Denial-of-Service.

DPD: Delegation Path Discovery.

DPV: Delegation Path Validation.

DSA: Directory System Agent.

DSS: Digital Signature Standard.

ECC: Elliptic Curve Cryptography.

ECDSA: Elliptic Curve Digital Signature Algorithm.

EE: End Entity.

FTP: File Transfer Protocol.

HTTP: HyperText Transfer Protocol.

IBAC: Identity Based Access Control.

IC: Identity Certificate.

I-CVT: Improved Certificate Verification Tree.

IETF: Internet Engineering Task Force.

IKE: Internet Key Exchange.

IP: Internet Protocol.

IPKI: Internet PKI.

ITU: International Telecommunication Union.

KDC: Key Distribution Center.

LBAC: Lattices Based Access Control.

LDAP: Lightweight Directory Access Protocol.

MAC: Message Authentication Code.

MAC: Mandatory Access Control.

MANET: Mobile Ad-hoc Network.

MD5: Message Digest 5.

ME: Mobile Entity.

MHT: Merkle Hash Tree.

MIME: Multipurpose Internet Mail Extensions.

NCA: Nested Certificate Authority.

NPKI: Nested certificate based PKI.

NSI: New Security Infrastructure.

NT: New Technology.

OASIS: Organization for the Advancement of Structured Information Standards.

OCSP: Online Certificate Status Protocol.

OLSR: Optimal Link State Routing.

OSF: Open Software Foundation.

OWF: One Way Function.

PAPI: Point of Access to Providers of Information.

PCA: Public Primary Certification Authority.

PDA: Personal Digital Assistant.

PEM: Privacy-Enhanced Electronic Mail.

PGP: Pretty Good Privacy.

PKC: Public Key Certificate.

PKI: Public Key Infrastructure.

PKIS: PKI Server.

PKIX: Public-Key Infrastructure X.509.

PMI: Privilege Management Infrastructure.

PoA: Point of Access.

PROSEARCH: PROtocol to Simplify the cErtification pAth discoveRy Constructing a Hierarchy.

PSSP: PKI Server-Server Protocol.

RA: Registration Authority.

RBAC: Role Based Access Control.

RCA: Root Certification Authority.

RFC: Request For Comments.

RPC: Remote Procedure Call.

RSA: Rivest Shamir Adleman.

RTS: Request To Send.

SAML: Security Assertion Markup Language.

SDSI: Simple Distributed Security Infrastructure.

SHA-1: Secure Hash Algorithm 1.

SHAR: SHibboleth Attribute Requester.

SIM: Subscriber Identity Module.

S/MIME: Secure Multipurpose Internet Mail Extensions.

SOA: Source Of Authority.

SOAP: Simple Object Access Protocol.

SPKI: Simple Public Key Infrastructure.

SSL: Secure Socket Layer.

SSO: Single Sign On.

TLS: Transport Layer Security.

TRUTHC: Trust Relationship Using Two Hash Chains.

TTP: Trusted Third Party.

URL: Universal Resource Locator.

UTC: Universal Time Coordinated.

VR: Verisign Root.

WAP: Wireless Application Protocol.

WIM: Wireless Identity Module.

WLAN: Wireless Local Area Network.

WML: Wireless Markup Language.

WPKI: WAP PKI.

WSG: WAP Security Group.

WTLS: Wireless Transport Layer Security.

W3C: World Wide Web Consortium.

XML: eXtensible Markup Language.

ZK: Zero Knowledge.

Capítulo 1

Introducción

1.1 Acerca de la Tesis 1
1.2 Motivación 2
1.3 Objetivos y Contribuciones de esta Tesis 4
1.4 Justificación 6
1.5 Organización de la Tesis 8
1.6 Publicaciones 9

1.1 Acerca de la Tesis

Esta tesis doctoral ha sido desarrollada en el Departamento de Ingeniería Telemática (ENTEL)[1] de la Universidad Politécnica de Cataluña[2]. La investigación desarrollada en esta tesis se enmarca dentro del proyecto, financiado por el Ministerio de Educación y Ciencia:

ARPA: Adquisición y Revocación de Privilegios en esquemas de Autorización
(CICYT-TIC2003-08184-C02-02)

[1] ENTEL: www-entel.upc.edu

[2] UPC: www.upc.edu

1.2 Motivación

Actualmente, Internet se presenta como un medio ideal para comercializar en forma automática una infinita variedad de productos y servicios a nivel global. Sin embargo, todo negocio, ya sea de persona a persona, telefónico o en este caso electrónico, se basa en una relación de confianza entre las partes. En los medios tradicionales de comercialización, esta confianza se logra, en la mayoría de los casos, a través de un conocimiento personalizado entre el vendedor y el comprador. Pero Internet, a pesar de ser un excelente medio de comunicación mundial, no permite el contacto directo de las partes. Por tanto, lo que por un lado es una de sus grandes ventajas, también puede resultar un gran impedimento. A través de Internet, los clientes pueden llegar a sentirse desprotegidos al no poder verificar personalmente que la empresa con la que están realizando una operación es legítima y verdadera, y que en caso de no estar satisfechos con el producto adquirido pueden devolverlo sin problemas [Com06]. Por ello, se hace necesario adoptar medidas de seguridad electrónica que nos permitan autenticar a nuestros compañeros de negocios, clientes y proveedores, antes de ocuparnos del intercambio de información, bienes y servicios [KLL02]. Esta necesidad de autenticación se evidencia también en otro tipo de actividades que se realizan hoy en día a través de Internet como gobierno electrónico, banca electrónica, educación virtual, etc.

La criptografía de clave pública o asimétrica ofrece una solución a este problema. Aquí, cada usuario posee un par de claves: la clave privada, conocida únicamente por su dueño, y la clave pública, que puede ser ampliamente distribuida. La información cifrada con la clave privada puede ser descifrada únicamente con la clave pública que le corresponde y viceversa [DH76]. Así, la criptografía asimétrica le permite a las partes verificar que la persona o entidad con la que están realizando una transacción es efectivamente quien envió los datos recibidos. Sin embargo, también necesitan saber con certeza qué clave pública pertenece a la entidad con la que se están comunicando. Por ello se creó la Infraestructura de Clave Pública (PKI – *Public Key Infrastructure*) [ITU00]. PKI cuenta con terceras partes de confianza (TTPs – *Trusted Third Parties*) conocidas como Autoridades de Certificación (CAs – *Certification Authorities*) y encargadas de expedir los llamados Certificados de Clave Pública

(PKCs – *Public Key Certificates*). Los PKCs son estructuras de datos que vinculan la identidad de cada usuario con una clave pública particular. Cada certificado es firmado digitalmente por la CA que lo emite y luego es puesto en un sitio público de fácil acceso (repositorio). Por tanto, la PKI se encarga de gestionar el uso y expedición de los PKCs.

Sin embargo, esta seguridad no es suficiente cuando se trata de controlar el acceso a los recursos y servicios que se ofrecen, principalmente, porque los sistemas de autenticación basados en identidad son demasiado rígidos y poco extensibles para reflejar los requerimientos que exigen actualmente las organizaciones. De este modo, se requiere reflejar algo más que la información personal del usuario, como su papel en la organización, su pertenencia a un grupo determinado, etc., para ofrecer mayor granularidad a la hora de establecer los permisos de control de acceso. Esto se hace gestionando el servicio de autorización. Los modelos de control de acceso basados en atributos especifican el acceso a un servicio teniendo en cuenta una colección de atributos. La naturaleza de estas colecciones y sus propiedades determinan la expresividad de las políticas [VSJ05]. Es el caso de la Infraestructura de Gestión de Privilegios (PMI – *Privilege Management Infrastructure*) [ITU00] que utiliza Certificados de Atributos (ACs – *Attribute Certificates*) para vincular un conjunto de privilegios con una entidad. Las TTPs que expiden y firman estos certificados se llaman Autoridades de Atributos (AAs – *Attribute Authorities*). De esta manera, un verificador puede autorizar el acceso a un recurso o servicio si los privilegios en el AC del usuario cumplen con la política de control de acceso especificada.

La autorización está íntimamente ligada a la autenticación en los modelos de control de acceso, por eso, para que exista una PMI debe preexistir una PKI.

Aunque organismos como el IETF (*Internet Engineering Task Force*) presentaron en los últimos años diferentes propuestas de estandarización que han contribuido a acercar la tecnología PKI a Internet (PKIX) [HPF99], el despliegue y uso de esta infraestructura no ha alcanzado las expectativas. Una razón por la que PKIX no ha sido completamente aceptado es debido a la complejidad del sistema. Cualquier aplicación que desee usar PKI debe implementar una lógica compleja para el análisis sintáctico de los certificados, la construcción de los caminos de certificación y la gestión de políticas. La construcción de caminos de

certificación contribuye particularmente a incrementar la complejidad del sistema, utilizando el método no estandarizado de búsqueda y recuperación de certificados. Así, muchas aplicaciones no utilizan o no pueden utilizar la tecnología de clave pública [Hun02].

1.3 Objetivos y Contribuciones de esta Tesis

Ya que los certificados se almacenan en repositorios no confiables, antes de fiarse en su contenido, los usuarios deben comprobar su integridad, autenticidad y vigencia. Esto se hace verificando la firma y el estado de validez de los mismos, mediante el proceso de validación.

Para verificar la firma de un certificado, el usuario necesita conocer la clave pública de la autoridad que expidió dicho certificado. Sin embargo, una PKI puede tener varias CAs, cada una con una clave pública distinta, y por lo general el usuario no confía en todas ellas. Así, para que el usuario pueda verificar la firma de los certificados expedidos por autoridades diferentes a sus CAs de confianza, debe existir cierta relación de confianza entre las autoridades de la PKI. Estas relaciones de confianza le permitirán al usuario formar cadenas de certificados a partir de las autoridades en que confía, conocidas como caminos de certificación.

En PMI, además de todo esto, tenemos los llamados caminos de delegación, ya que las AAs pueden delegar sus privilegios a otras entidades. La validación de estos caminos involucra a su vez la validación de los caminos de certificación.

El proceso de validación de caminos de certificación [HPF02] y delegación [PH02] es bastante complejo pues involucra: descubrir los caminos, recuperar los certificados que hacen parte de ellos, verificar la firma digital de cada uno de estos certificados, constatar que ninguno haya expirado o haya sido revocado; y en el caso de los caminos de delegación también implica verificar que cada autoridad tiene los privilegios y autorización suficientes para delegar sus atributos y acceder al recurso indicado. Por tanto, uno de los retos al que deben enfrentarse las arquitecturas PKI y PMI es la gestión eficiente de estos caminos [CG03].

El problema que pretende resolver esta tesis doctoral es: *¿Cómo simplificar el proceso de validación de caminos desde el punto de vista del verificador?*, por lo que el objetivo es proponer diferentes mecanismos que contribuyan a mejorar el desempeño del proceso de validación de certificados en las arquitecturas de autenticación y autorización.

La complejidad del proceso de validación de caminos, nos lleva a preguntarnos si un verificador con capacidades limitadas puede realizar correctamente este proceso. Por ello, en esta tesis se evalúan el coste computacional y la capacidad de almacenamiento que requieren los verificadores. Además, describimos las distintas propuestas, encontradas en la literatura, que contribuyen a mejorar la eficiencia de proceso de validación de caminos.

La verificación del estado de revocación de cada certificado es un aspecto ampliamente estudiado [HPF02] [MAM99] [Coo99] [MFE03], sin embargo, la construcción de los caminos y la verificación de firma de los certificados, que consumen gran cantidad de tiempo y le exigen al verificador determinadas capacidades de cálculo y almacenamiento, todavía no han sido suficientemente explorados. En la presente tesis doctoral proponemos dos mecanismos que contribuyen a mejorar la eficiencia de estos dos procedimientos: PROSEARCH y TRUTHC, respectivamente.

Ya que el proceso de validación de caminos debe ser rápido, eficiente y apropiado para grandes infraestructuras, la reducción del número de operaciones que representan mayor carga computacional para los verificadores contribuye a agilizar este proceso. Por esta razón, hemos propuesto TRUTHC (*Trust Relationship Using Two Hash Chains*), que reduce el número de operaciones de cifrado con clave pública que debe realizar un verificador durante el proceso de comprobación de las firmas digitales, sustituyéndolas por simples operaciones de hash.

Por otra parte, la existencia de relaciones de confianza bidireccionales entre las entidades de una PKI dificulta el proceso de construcción de caminos de certificación, puesto que todas las opciones no conducen a la entidad objetivo y pueden existir múltiples caminos entre dos entidades [PH00]. PROSEARCH (*Protocol to Simplify the Certification Path Discovery Constructing a Hierarchy*) permite construir una jerarquía virtual entre las diferentes entidades de una PKI híbrida o en malla. De esta manera, se establecen relaciones de

confianza unidireccionales entre las entidades, que le facilitan el proceso de construcción de caminos al verificador.

La funcionalidad y eficiencia de estos mecanismos es evaluada, así como su interoperabilidad con los estándares de seguridad actuales.

También se especifica cómo pueden ser usados TRUTHC y PROSEARCH en el servicio de autorización.

1.4 Justificación

El mercado de PKI ha ido creciendo gradualmente en los últimos años. Para 2002, los analistas cifraron en alrededor de 54 millones de euros el volumen del mercado español de servicios de certificación, con las administraciones públicas y el sector financiero, seguidos del sector de las telecomunicaciones, los colegios profesionales y algunas empresas de servicios como sus principales clientes. Evidentemente, su expansión aparece estrechamente ligada al grado de penetración de Internet y al desarrollo del comercio electrónico. En general, según Datamonitor, el mercado mundial de PKI en 2001 supuso 738 millones de euros, y en 2002 alcanzó los 1.106 millones [MM02].

De acuerdo a Safelayer, empresa líder del mercado PKI en España, la puesta en marcha del DNI Digital va a suponer un efecto multiplicador en cuanto a la extensión de la firma digital en la sociedad. Según prevé esta empresa, aunque el sector empresarial será el primero en extender este uso, no se debe perder de vista su extensión en el usuario personal, tanto profesionales liberales como usuarios particulares [Saf05].

También se prevé que la PKI que ha desarrollado SAFELAYER, soportará hasta 60 millones de usuarios con al menos dos certificados por cada uno de ellos. En estos 60 millones de usuarios, se integran los 33 millones de usuarios actuales que contarán con el nuevo DNI electrónico, más el crecimiento vegetativo calculado para los próximos quince años, que según estima el Ministerio del Interior español, será de un millón de usuarios nuevos por año [Gig05].

Dado el crecimiento inminente del mercado PKI, se hace necesario solucionar los diferentes problemas que aún presenta esta tecnología en áreas como la interoperabilidad de las PKIs de diferentes organizaciones y la complejidad del sistema.

La validación de certificados es parte fundamental de PKI y PMI pues permite constatar la veracidad de la información que se comparte. El proceso de validación es complejo especialmente cuando crecen las infraestructuras de autenticación y autorización creando cadenas de certificados de longitud considerable. Esto dificulta la construcción de caminos puesto que pueden verse involucradas diferentes autoridades y dominios de confianza, siendo necesario un acuerdo o correspondencia entre sus políticas. Además, ciertas arquitecturas de seguridad no han sido diseñadas para ser escalables, lo que introduce mayores dificultades en el establecimiento de los caminos.

Ya que las relaciones de confianza unidireccionales facilitan la construcción de los caminos de certificación, PROSEARCH ha sido diseñado para establecer un modelo de confianza jerárquico en una PKI con relaciones de confianza bidireccionales entre sus entidades, disminuyendo el tiempo de búsqueda de caminos por parte del verificador e incrementando la eficiencia del proceso de validación.

La validación de caminos largos puede ser también compleja para el verificador, desde el punto de vista computacional, puesto que los algoritmos de clave pública requieren de cálculos matemáticos complejos; como considerando el conjunto de recursos necesarios para obtener los certificados, almacenarlos y verificarlos. Así, dispositivos con capacidades limitadas, como los teléfonos móviles, tarjetas inteligentes y PDAs, pueden no disponer de recursos suficientes para llevar a cabo este proceso.

TRUTHC le permite al verificador chequear la integridad y el origen de los certificados realizando únicamente operaciones de hash, lo que reduce el número de operaciones de descifrado con clave pública necesarias durante el proceso de validación de caminos e incrementa su eficiencia, al disminuir el coste computacional de las operaciones criptográficas.

1.5 Organización de la Tesis

Esta tesis doctoral esta estructurada de la siguiente manera:

- El capítulo 2, "Validación en Sistemas de Autenticación y Autorización", contiene una clasificación de los diferentes métodos de autenticación y autorización, centrándose principalmente en las características de PKI. Además, describe las operaciones que se deben llevar a cabo para validar caminos de certificación y de delegación, así como los mecanismos más utilizados para realizar estos procesos.
- En el capítulo 3, "Coste Computacional de Validación de Caminos de Certificación y Delegación" se evalúa el coste computacional requerido para validar caminos de certificación cuando el verificador dispone de un terminal móvil de poca capacidad y se utilizan los mecanismos de revocación estándar. También se evalúa el coste computacional de los caminos de delegación.
- El capítulo 4 "Capacidad de Almacenamiento en Validación de Caminos de Certificación", determina la capacidad de almacenamiento que debe tener un verificador para validar caminos de certificación, con y sin revocación.
- En el capítulo 5, "TRUTHC (Trust Relationship Using Two Hash Chains)" presentamos un método para disminuir la carga computacional del verificador durante el proceso de validación de caminos de certificación. THUTHC utiliza cadenas de hash para establecer una relación de confianza alternativa entre las entidades de una PKI jerárquica.
- El capitulo 6, "PROSEARCH (Protocol to Simplify the Certification Path Discovery Constructing a Hierarchy)", propone un protocolo para disminuir el tiempo de búsqueda de caminos de certificación, creando una jerarquía virtual entre las diferentes entidades que la conforman una PKI con arquitectura en malla.
- En el capítulo 7, "Evaluación de PROSEARCH en Diferentes Entornos", se evalúa el desempeño de PROSEARCH en infraestructuras críticas y redes ad-hoc.

- El capítulo 8 “Conclusiones y Líneas Futuras”, enumera los resultados y contribuciones de esta tesis y propone líneas futuras de investigación.

1.6 Publicaciones

En este apartado se relacionan las publicaciones a que ha dado lugar esta tesis, en orden ascendente por su fecha de publicación.

1.6.1 Congresos

1.6.1.1 Nacionales

- Cristina Satizábal, Jordi Forné. “Revocación de Certificados en la Validación de Caminos de Certificación”. *En VIII Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2004)*. Leganés (España). Septiembre, 2004. pp. 605 – 614. ISBN: 84-7978-650-7.
- Cristina Satizábal, Rafael Páez, Jordi Forné. “Construyendo Caminos de Certificación Mediante Cadenas de Hash”. *En V Jornadas de Ingeniería Telemática (JITEL 2005)*. Vigo (España). Septiembre, 2005. pp. 343-350. ISBN: 84-8408-346-2.

1.6.1.2 Internacionales

- Cristina Satizábal, Rafael Páez, Jordi Forné. “Certification Path Validation in WPKI”. *En Infocom 2005 Student Workshop*. Miami (USA). Marzo, 2005.
- Cristina Satizábal, Rafael Páez, Jordi Forné. “Privilege Delegation: Path Validation vs. Certificate Revocation”. *En IV Congreso Internacional de Electrónica y Tecnologías de Avanzada*. Pamplona (Colombia). Marzo, 2005. ISBN: 958-97105-7-3.
- Cristina Satizábal, Rafael Páez, Jordi Forné. “PKI Trust Relationship Using Hash Chains”. *En International Conference on Advances in the Internet, Processing,*

Systems and Interdisciplinary Research (IPSI 2005). Carcassonne (Francia). Abril, 2005. ISBN: 86-7466-117-3.

- Cristina Satizábal, Rafael Páez, Jordi Forné. "WAP PKI and Certification Path Validation". *En 11th Open European Summer School: Networked Applications (EUNICE 2005)*. Colmenarejo (España). Julio, 2005. pp. 185-190. ISBN: 84-89315-43-4.
- Cristina Satizábal, Rafael Páez, Jordi Forné. "PKI Trust Relationships: from a Hybrid Architecture to a Hierarchical Model". *En First International Conference on Availability, Reliability and Security (ARES 2006)*. Vienna (Austria). Abril, 2006. pp. 563-570. ISBN: 978-0-7695-2567-9. Publicado por IEEE Computer Society Press.
- Cristina Satizábal, Rafael Páez, Jordi Forné. "PROSEARCH: A Protocol to Simplify Path Discovery in Critical Scenarios". *En* 1st *International Workshop on Critical Information Infrastructures Security (CRITIS'06)*. Samos Island (Grecia). Septiembre, 2006. Editado por Springer en la serie Lecture Note in Computer Science, vol. 4347, pp.151-165.
- Cristina Satizábal, Jordi Forné, Juan Hernández-Serrano, Josep Pegueroles. "Building Hierarchical Public Key Infrastructures in Mobile Ad-Hoc Networks". *En 2nd International Conference on Mobile Ad-hoc and Sensor Networks (MSN 2006)*. Hong Kong (China). Diciembre, 2006. Editado por Springer en la serie Lecture Note in Computer Science, vol.4325, pp.485-496.

1.6.2 Revistas

1.6.2.1 Nacionales

- Cristina Satizábal, Rafael Páez, Jordi Forné. "Privilege Delegation: Path Validation vs. Certificate Revocation". *Revista Colombiana de Tecnologías de Avanzada*. Vol. 1(5), pp. 8-14. 2005. ISSN: 1692-7257.

1.6.2.2 Internacionales

- Cristina Satizábal, Rafael Páez, Jordi Forné. "Building a Virtual Hierarchy for Managing Trust Relationships in a Hybrid Architecture". *Journal of Computers (JCP)*. Vol. 1(7), pp. 60-68, 2006. ISSN1796-203X.
- Cristina Satizábal, Rafael Páez, Jordi Forné. "WAP PKI and Certification Path Validation". *International Journal of Internet Protocol Technology (IJIPT)*. Vol. 1(2), 2007. ISSN (Online): 1743-8217 - ISSN (Print): 1743-8209 (Aceptado).
- Cristina Satizábal, Juan Hernández-Serrano, Jordi Forné, Josep Pegueroles. "Building a Virtual Hierarchy to Simplify Certification Path Discovery in Mobile Ad-Hoc Networks". Computer Communications (Aceptado).

1.6.3 Otras Publicaciones

Existen además algunas publicaciones colaterales a este trabajo de tesis:

- Rafael Páez, Cristina Satizábal, Jordi Forné. "Cooperative Agents and Software Fingerprinting for Security in Intrusion Detection Systems Based on Autonomous Agents". *En IASTED International Conference on Databases and Applications (DBA 2005)*. Innsbruck, Austria. Febrero, 2005. ISSN: 1027-2666; ISBN: 0-88986-460-8.
- Rafael Páez, Cristina Satizábal, Jordi Forné. "Analysis of Intrusion Detection Systems Based on Autonomous Agents and their Security". *En International Conference on Advances in the Internet, Processing, Systems and Interdisciplinary Research (IPSI 2005)*. Carcassonne (Francia). Abril, 2005. ISBN: 86-7466-117-3.
- Rafael Páez, Cristina Satizábal, Jordi Forné. "Agentes Cooperativos y Software Fingerprinting para la Seguridad en Sistemas de Detección de Intrusos Basados en Agentes Autónomos" *En XV Jornadas Telecom I+D 2005*. Madrid (España). Noviembre, 2005. ISBN: 84-689-3794-0
- Rafael Páez, Cristina Satizábal, Jordi Forné. "Cooperative Itinerant Agents (CIA): Security Scheme for Intrusion Detection Systems". *En International Conference on*

Internet Surveillance and Protection (ICISP 2006). Cap Esterel, Francia. Agosto, 2006. ISBN: 0-7695-2649-7.

Capítulo 2

Validación en Sistemas de Autenticación y Autorización

2.1 Introducción 13
2.2 Criptografía 14
2.3 Autenticación 17
2.4 Autorización 33
2.5 Validación de Certificados 45
2.6 Conclusiones 63

2.1 Introducción

Un sistema de control de acceso protege la confidencialidad, integridad y disponibilidad de los recursos mediante mecanismos que dificultan la entrada de entidades no autorizadas a los mismos. El control de acceso consta generalmente de dos pasos:

- En primer lugar, la *autenticación*, que identifica al usuario o a la máquina que trata de acceder a los recursos.
- En segundo lugar, la cesión de derechos, es decir, la *autorización*, que dota al usuario de privilegios para poder efectuar ciertas operaciones con los datos protegidos, tales como leerlos, modificarlos, etc.

PKI y PMI utilizan certificados digitales para vincular las claves públicas y los privilegios con sus propietarios, brindando los servicios de autenticación y autorización, respectivamente.

Ya que los certificados se almacenan en repositorios no confiables y su tiempo de validez es limitado, antes de hacer uso de ellos, los usuarios deben comprobar que contienen información válida. Nace así el proceso de validación, mediante el cual se verifica la firma y vigencia de los certificados. Éste es un proceso complejo, especialmente cuando involucra diversas autoridades y políticas de certificación.

El objetivo principal de este capítulo es dar una visión global de los mecanismos de autenticación y autorización existentes, así como describir el proceso de validación de caminos de certificación y de delegación en PKI y PMI, respectivamente. En la sección 2.2 se definen las técnicas criptográficas más utilizadas. La sección 2.3 contiene una clasificación de los métodos de autenticación existentes, centrándose en los componentes y funciones de PKI, así como en los estándares relacionados con esta infraestructura. La sección 2.4, clasifica los mecanismos de control de acceso y describe los estándares y soluciones de autorización empleadas en entornos web. En la sección 2.5 describimos los diferentes pasos que conforman el proceso de validación de caminos de certificación y de delegación, así como los mecanismos propuestos para llevarlos a cabo. Finalmente, la sección 2.6 concluye.

2.2 Criptografía

Para proveer de manera adecuada los servicios de autenticación y autorización se emplean los siguientes mecanismos criptográficos:

- ***Criptografía Simétrica***: Es llamada algunas veces criptografía de clave secreta porque el emisor y el receptor comparten una clave secreta, utilizada para cifrar y descifrar los mensajes que intercambian.

  Formalmente, se usa el algoritmo simétrico S para cifrar el mensaje M usando la clave secreta K.

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

  Aplicando el algoritmo inverso S^{-1} se descifra el mensaje secreto C usando la clave secreta K

$$M=S^{-1}_{K}(C) \quad (2.2)$$

DES (*Digital Encryption Standard*) [NIS88] y su versión extendida Triple-DES (3DES) [NIS99] han sido los sistemas de criptografía simétrica más populares durante muchos años. Recientemente, AES (*Advanced Encryption Standard*) [NIS01] se ha diseñado como sucesor del DES.

En general, los sistemas de clave simétrica son más simples y rápidos que los de clave pública, pero su principal inconveniente es que las dos partes deben intercambiar la clave simétrica de manera segura. Esto es importante en muchos escenarios y se conoce como *problema de distribución de claves*.

- ***Criptografía de Clave Pública:*** Es también conocida como criptografía asimétrica puesto que involucra el uso de dos claves, en contraste con la criptografía simétrica que usa sólo una clave. Una de estas claves es pública (es decir, conocida por todos) y la otra es privada (es decir, conocida únicamente por su dueño). La criptografía asimétrica facilita la distribución de claves ya que la clave pública no necesita ser mantenida en secreto, y puede distribuirse libremente, mientras la clave privada nunca se transmite.

  Diffie y Hellman [DH76] postularon las condiciones que debe cumplir un sistema de clave pública:

  - Es computacionalmente fácil para el usuario B generar un par de claves: clave pública PK_B y clave privada SK_B.
  - Es computacionalmente fácil para un emisor A, conociendo la clave pública PK_B y el mensaje a ser cifrado M, generar el texto cifrado C, aplicando el algoritmo asimétrico de cifrado E.

    $$C = E_{PKB}(M) \quad (2.3)$$

  - Es computacionalmente fácil para el receptor B descifrar el texto cifrado C usando su clave privada SK_B y el algoritmo asimétrico de descifrado D para recuperar el mensaje original M.

    $$M = D_{SKB}(C) = D_{SKB}(E_{PKB}(M)) \quad (2.4)$$

  - Es computacionalmente imposible para un oponente, que conozca la clave pública de B, determinar la clave privada SK_B.

- Es computacionalmente imposible para un oponente, que conozca la clave pública PK_B y el texto cifrado C, recuperar el mensaje original M.

Los sistemas de clave pública se usan principalmente para los siguientes propósitos:

- *Cifrado:* El emisor cifra un mensaje con la clave pública del receptor.
- *Firma Digital:* Se genera una prueba digital que sólo el emisor de un mensaje puede crear, pero que todos pueden identificar como perteneciente a ese emisor. Para ello, el emisor cifra el mensaje con su clave privada.
- *Intercambio de Claves:* Dos partes cooperan para intercambiar una clave de sesión (clave simétrica). Existen diferentes alternativas, que involucran la clave privada de una o ambas partes.

El algoritmo de clave pública más famoso es RSA (Rivest Shamir Adleman) [RSA78], que puede ser usado para las tres aplicaciones, mientras DSS (*Digital Signature Standard*) [NIS00] es utilizado ampliamente sólo en firmas digitales y DH (Diffie Hellman) [DH76] únicamente en el intercambio de claves.

- ***Funciones de Hash:*** Una función de hash H es una función unidireccional (OWF – *One Way Function*), que toma una entrada de longitud variable M (pre-imagen) y calcula una cadena de salida m de longitud fija - usualmente más pequeña que la entrada - conocida como valor de hash, resumen o valor de chequeo.

  Que sea una función unidireccional quiere decir que:

  - Dado un mensaje M es computacionalmente fácil calcular $m = H(M)$.
  - Dado un valor de hash m es computacionalmente imposible encontrar un mensaje M tal que $H(M) = m$.

  Además, una función de hash H puede ser libre de colisión, lo que significa que dado un mensaje M es imposible encontrar otro mensaje $M' \neq M$ tal que $H(M')=H(M)$.

  En criptografía, las funciones de hash se usan comúnmente en el cálculo de firmas digitales. Ya que las funciones de hash son más rápidas que los algoritmos de firma

digital, se suele calcular la firma digital sobre el valor de hash del documento a firmar, que generalmente es más pequeño.

Ejemplos bien conocidos de funciones de hash son MD5 (*Message Digest* 5) [Riv92] y SHA-1(*Secure Hash Algorithm*-1) [NIS95].

2.3 Autenticación

Por autenticación se entiende cualquier método que permite comprobar de manera segura cierta característica de un objeto, como su origen, su no manipulación, su identidad, etc. [Luc99]. Existen dos grandes grupos dentro de los métodos de autenticación:

- ***Autenticación de Mensaje:*** Garantiza la procedencia de un mensaje, identificando la entidad que lo emitió. Por definición, este tipo de autenticación involucra también la integridad de los datos, es decir, permite detectar si el contenido del mensaje fue accidental o maliciosamente alterado.
- ***Autenticación de Usuario:*** Garantiza que un usuario es quien afirma ser. En este proceso se corrobora la identidad del usuario en tiempo real, es decir, en el mismo momento en que se están comunicando el verificador y el usuario.

2.3.1 Autenticación de Mensaje

Las técnicas más utilizadas para proveer autenticación de mensaje, según [MOV97], son:

- ***MAC (Message Authentication Code)***: Los códigos de autenticación de mensajes fueron diseñados especialmente para aplicaciones donde se requiere la integridad de los datos (pero no necesariamente su privacidad). Un MAC es un tipo de función unidireccional que toma dos entradas (un mensaje y una clave secreta) y retorna una cadena de tamaño fijo, siendo imposible volver a obtener la misma salida sin conocer la clave. Así, el emisor de un mensaje *M*, usa la clave secreta compartida *K* para

calcular un MAC $h_K(M)$ sobre el mensaje y envía al receptor M y $h_K(M)$. El receptor determina la identidad de la fuente del mensaje (utilizando, por ejemplo, un campo identificador en el texto no cifrado) y separa el MAC de los datos. Luego, calcula un MAC sobre estos datos, utilizando la clave MAC compartida, y compara el MAC calculado con el recibido. Si estos valores coinciden, el receptor interpreta que los datos son auténticos y que no fueron modificados durante su transmisión.

- ***Firma Digital**:* Simula una firma real. Para ello, se genera una prueba digital que sólo el emisor de un mensaje puede crear, pero que todos pueden identificar como perteneciente a ese emisor. Un cifrado utilizando la clave privada del emisor sirve como firma, porque sólo el propietario de la clave privada puede crearlo y todos pueden verificarlo con la clave pública correspondiente. El cifrado (firma) puede aplicarse a todo el mensaje o a un pequeño bloque de datos que sea función del mensaje, por ejemplo, el valor de hash. Por tanto, el emisor A envía al receptor B el mensaje M y la firma calculada sobre el hash del mensaje $E_{SK_A}(H(M))$, donde SK_A es la clave privada del emisor, E es el algoritmo de cifrado asimétrico utilizado y $H(M)$ es el hash del mensaje. El receptor debe entonces obtener la clave pública del emisor PK_A y realizar una operación de cifrado sobre la firma del mensaje, con lo que obtendría $H'(M)$. Posteriormente, B debe calcular el hash del mensaje y compararlo con $H'(M)$. Si los dos resultados coinciden, se corrobora la autenticidad y la integridad del mensaje.

2.3.2 Autenticación de Usuario

Las técnicas de autenticación de usuario pueden clasificarse en tres categorías, según [MOV97], dependiendo de qué deba presentar el usuario para demostrar su identidad:

- ***Algo Conocido:*** El usuario debe demostrar que conoce algún tipo de información secreta, por ejemplo, una palabra clave, un número de identificación personal (PIN), una clave privada, etc.

- ***Algo que Posee:*** El usuario requiere para identificarse algún tipo de dispositivo, como: una tarjeta magnética, una tarjeta inteligente, un generador de contraseñas, una llave electrónica, etc.
- ***Algo Inherente (Propio del Individuo):*** Aquí se requiere alguna característica fisiológica del usuario (biometría), por ejemplo: las huellas digitales, la voz, la retina, el iris, etc. Esta categoría puede considerarse como un caso particular de la anterior, donde el dispositivo es el propio usuario.

Las técnicas de autenticación de usuario basadas en algo conocido, se pueden clasificar a su vez en tres categorías de acuerdo al nivel de seguridad que ofrecen, según se especifica en [MOV97]:

- ***Autenticación Débil:*** Es el tipo de autenticación más extendido, y al que los usuarios ya están acostumbrados. Aquí, una contraseña o palabra clave (*password*), asociada a cada usuario, es el secreto compartido entre el usuario y el sistema. Para identificarse, el usuario introduce un identificador (*login*) y su contraseña. El sistema comprueba entonces que esa contraseña coincida con la almacenada para dicho identificador.

  Ya que las contraseñas son fijas y reutilizables, si un atacante se entera de la clave de un usuario, puede suplantar su identidad fácilmente.

  Para protegerse de los posibles ataques por fuerza bruta, de los que pueden ser víctimas estos sistemas, se limita el número de intentos que el usuario puede hacer desde un terminal concreto y se introducen retardos cuando la contraseña introducida es errónea.
- ***Autenticación Fuerte:*** En este tipo de autenticación una entidad (el demandante) prueba su identidad ante otra entidad (el verificador) demostrando que conoce un secreto asociado con su identidad, pero sin revelar dicho secreto al verificador. Esto se hace respondiendo a un desafío variable. Dicha respuesta depende tanto del secreto del demandante como del desafío. El desafío es típicamente un número elegido por el verificador (aleatoria y secretamente) al inicio del protocolo. Si un atacante obtiene la

respuesta al desafío de un usuario determinado, esta información no le será útil para suplantar posteriormente a ese usuario, ya que el desafío será diferente.

Los mecanismos de autenticación fuerte basados en criptografía simétrica, requieren que el demandante y el verificador compartan una clave secreta. Cuando el sistema cuenta con un gran número de usuarios se suele usar un servidor confiable en línea, con el que cada usuario comparte una clave. Este servidor actúa a manera de hub, proporcionando una clave de sesión común a las dos partes comunicantes cada vez que quieran autenticarse una con la otra. El protocolo Kerberos provee autenticación fuerte basada en cifrado simétrico e involucra el uso de una tercera parte de confianza en línea.

En los mecanismos de autenticación fuerte basados en criptografía asimétrica, el demandante demuestra que posee su clave privada, bien sea descifrando un desafío cifrado con su clave pública o firmando digitalmente el desafío. Para no comprometer la seguridad del sistema, el par de claves usado por estos mecanismos no debe ser usado con otros propósitos. La Infraestructura de Clave Pública (PKI) utiliza autenticación fuerte basada en criptografía asimétrica para proveer sus servicios de seguridad.

- ***Autenticación de Conocimiento Nulo:*** Los protocolos de conocimiento nulo (ZK: *Zero-knowledge*) le permiten a un probador (el demandante) demostrar el conocimiento de un secreto sin revelar ninguna información útil para el verificador (más allá de la que el verificador es capaz de deducir antes de ejecutar el protocolo). Los sistemas de prueba interactivos son un ejemplo de este tipo de protocolos. Aquí, el objetivo del probador es convencer al verificador de que posee algún secreto, contestando correctamente preguntas que requieren del conocimiento de ese secreto para ser respondidas. Estos protocolos emplean técnicas asimétricas pero no utilizan firmas digitales o cifrados con clave pública. Tampoco usan cifradores de bloques, números de secuencia ni sellos de tiempo (*timestamps*).

2.3.3 Arquitecturas de Autenticación

Las técnicas de autenticación descritas anteriormente implican algún tipo de clave o secreto compartido entre las partes comunicantes. Debe existir, por tanto, una infraestructura de seguridad que se encargue de la generación, distribución y gestión de los secretos. Esta infraestructura de seguridad brindará una base segura a toda la organización y deberá ser accesible por todas las aplicaciones y objetos de la organización que necesiten seguridad [AL03].

En una infraestructura de autenticación, una tercera parte de confianza o autoridad de autenticación proporciona el servicio de autenticación a los usuarios. Cada autoridad de autenticación gobierna un conjunto de recursos localizados en dispositivos que hacen parte de su área de actuación. Esta área de actuación es llamada *dominio* (en la terminología NT), *reino* (en la terminología Kerberos) o *celda* (en la terminología DCE). Para autenticarse, el usuario comparte una *credencial* (secreto) con la autoridad de autenticación. La autoridad de autenticación guarda la copia de esta credencial en una base de datos segura. Dependiendo del tipo de credenciales, el usuario puede simplemente recordarlas o guardarlas de alguna manera (por ejemplo, en una tarjeta inteligente).

Durante un proceso típico de autenticación, el usuario presenta su credencial (por ejemplo: *login* y *password*) o el resultado de una operación criptográfica que involucre su credencial, a la autoridad de autenticación. La autoridad de autenticación valida la credencial usando los datos guardados en su base de datos. Si la credencial proporcionada por el usuario y la guardada en la base de datos coinciden, o si el resultado de una operación criptográfica sobre la credencial guardada en la base de datos es igual a la información suministrada por el usuario, la identidad del usuario es considerada auténtica [Cle02].

Sin embargo, el usuario debe compartir una credencial con cada dominio y autenticarse cada vez que quiera acceder a algún recurso. Dada la diversidad de plataformas de computación, sistemas operativos y software de control de acceso, lo más deseable es registrarse en múltiples sistemas una vez y simultáneamente a través de una sola transacción. Nace así el registro único (SSO – *Single Sign-On*). Con SSO, el usuario debe autenticarse sólo

una vez para acceder a múltiples sistemas. El registro único puede usarse en infraestructuras con diversas autoridades de autenticación, es decir, implementadas en diferentes plataformas y gobernadas por diversas organizaciones. Las arquitecturas más simples son las que usan un solo conjunto de credenciales. Las dos más importantes, según [Cle02], son:

- ***Sistemas Basados en Testigos (Token-Based):*** En una arquitectura basada en testigos, los usuarios obtienen un testigo temporal después de autenticarse exitosamente ante su TTP. Este testigo puede ser guardado en el dispositivo del usuario y reutilizado para probar su identidad ante las TTPs de dominios de autenticación secundarios. Para validar el testigo de un usuario, estas TTPs usan métodos criptográficos basados en las claves secretas establecidas previamente entre ellas y la TTP del dominio de autenticación primario. Estas claves criptográficas permiten establecer una relación de confianza entre los diferentes dominios de autenticación.

  Kerberos es un estándar abierto definido por el IETF que ha sido implementado en diferentes plataformas. Aquí, los usuarios se identifican ante un servicio de autenticación central, llamado centro de distribución de claves (KDC – *Key Distribution Center*), que es la autoridad de autenticación. Si sus credenciales de autenticación son válidas, reciben un tiquete (*ticket*), que corresponde al testigo. Este tiquete le permite al usuario solicitar más tiquetes al KDC para acceder a otros recursos en el dominio de autenticación primario y los secundarios. Los tiquetes le sirven a los servidores de recursos para probar que el usuario ha sido autenticado antes por un servicio de autenticación confiable (el KDC).

  El protocolo de autenticación Kerberos inspiró el desarrollo, por parte de la OSF (*Open Software Foundation*), de los servicios de autenticación de DCE (*Distributed Computing Environment*). Microsoft implementa Kerberos como el protocolo de autenticación por defecto de Windows 2000.

  El sistema basado en testigos de Kerberos usa típicamente llamadas a procedimientos remotos (RPCs – *Remote Procedure Calls*) para transportar los tiquetes de autenticación.

- ***Sistemas Basados en PKI (PKI-Based):*** En esta arquitectura, los usuarios se registran primero ante una autoridad de autenticación confiable, la CA. Durante el proceso de registro los usuarios se identifican a través de un conjunto de credenciales. El software del usuario genera un par asimétrico de claves, y la clave pública es ofrecida a la CA para su certificación. Después de recibir las credenciales del usuario y la clave pública, la CA verifica si las credenciales son válidas. Si es así, genera un certificado de clave pública y se lo retorna al usuario. Este certificado y la clave privada son guardados de manera segura en el dispositivo del usuario u otro medio, como una tarjeta inteligente. Estos son usados para probar la identidad del usuario ante otras autoridades de certificación en solicitudes de autenticación posteriores. En esta arquitectura, la relación de confianza entre la autoridad de certificación primaria y las secundarias se establece a través de un certificado expedido por la CA primaria a la CA secundaria.

  Esta arquitectura es relativamente nueva, comparada con los sistemas basados en testigos. Sin embargo, en los últimos años, la industria del software en seguridad y organizaciones de estandarización, como el IETF, han hecho importantes esfuerzos para adaptar los sistemas basados en PKI al ambiente empresarial. Además, la mayoría de los proveedores de software han modificado sus aplicaciones para hacerlas compatibles con las credenciales utilizadas en PKI.

2.3.4 Infraestructura de Clave Pública (PKI)

En 1978, Kohnfelder [Koh78] planteó el uso de estructuras de datos llamadas certificados, firmados digitalmente por una TTP, para posibilitar el transporte seguro de las claves públicas. Actualmente, existen diversos tipos de certificados, pero los certificados de clave pública son quizás los más ampliamente usados.

Los PKCs, también llamados certificados de identidad (ICs – *Identity Certificates*), son estructuras de datos que vinculan una clave pública con la identidad de su propietario. Ya que estos van firmados digitalmente por la CA que los expide, pueden ser colocados en repositorios no confiables para que todos los usuarios del sistema tengan acceso a ellos.

Dada la popularidad de estos certificados, su formato se ha estandarizado en la recomendación X.509 [ITU00]. Todos los certificados X.509 contienen los siguientes campos, además de la firma:

- ***version:*** Identifica la versión del certificado según el estándar X.509. Existen tres versiones, cuyas diferencias serán especificadas más adelante.
- ***serialNumber:*** Es un número entero asignado por la CA a cada certificado. El valor del número serial tendrá que ser único para cada certificado expedido por una determinada CA (es decir, el nombre de la CA junto con el número de serie identificarán a un único certificado).
- ***signature:*** Identifica el algoritmo y función de hash utilizados por la CA para firmar el certificado (por ejemplo, md5WithRSAEncryption, sha1WithRSAEncryption, id-dsa-with-sha1, etc.).
- ***issuer:*** Identifica a la entidad que ha firmado y expedido el certificado, es decir, la CA.
- ***validity:*** Cada certificado es válido sólo durante un intervalo de tiempo, es decir, no es válido antes de la fecha de activación (*not-valid-before*), ni después de la fecha de expiración (*not-valid-after*).
- ***subject:*** Identifica al propietario de la clave pública del certificado.
- ***subjectPublicKeyInfo:*** Se utiliza para indicar la clave pública que se está certificando y para identificar el algoritmo con el que se creó dicha clave (por ejemplo, rsaEncryption, id-dsa, etc.).

Actualmente, existen tres versiones de certificados X.509. La versión 1 (v1) ha estado disponible desde 1988, por lo que es ampliamente usada y más genérica. La versión 2 (v2) introdujo el concepto de identificadores únicos de sujeto (*subjectUniqueIdentifier*) y emisor (*issuerUniqueIdentifier*). Se utilizan para brindar la posibilidad de reutilizar los nombres del sujeto y/o el emisor, sin embargo, la mayoría de los documentos sobre certificación recomiendan que los nombres no sean reutilizados. Por esta razón, estos certificados son poco usados. La versión 3 (v3) es la más reciente (1996) e incluye el campo *extensions*, que

permite la adición de nueva información a la estructura del certificado sin modificar su definición inicial. Una extensión está constituida por un identificador de extensión (*extnId*), una bandera de criticidad (*critical*) y un valor específico para la extensión identificada (*extnValue*). Cuando una aplicación está procesando un certificado y no reconoce una extensión, si la bandera de criticidad es *falsa*, puede ignorar dicha extensión. Si la bandera de criticidad es *verdadera*, las extensiones no reconocidas harán que la estructura se considere no válida, es decir, una extensión crítica no reconocida provocará el fracaso de la validación de dicho certificado. Cuando la aplicación reconoce y es capaz de procesar una extensión, la procesará independientemente del valor de su bandera de criticidad.

Para hacer uso de los certificados digitales, no sólo se necesita una CA, sino también una infraestructura que asegure la validez de las transacciones electrónicas que se realizan con dichos certificados. En [Shi00], una PKI se define como:

> "*El conjunto de hardware, software, gente, políticas y procedimientos necesario para crear, gestionar, guardar, distribuir y revocar certificados digitales, basado en criptografía asimétrica*"

La recomendación X.509 [ITU00] describe un marco de referencia para PKI, donde se define la sintaxis de los objetos, como certificados y listas de revocación de certificados (CRLs – *Certificate Revocation Lists*), pero no da las bases para la implementación de este tipo de infraestructuras. Por ello, se ha creado un conjunto de perfiles, incompatibles entre sí, que definen como la presencia y valores de estos objetos afectan la gestión de los certificados. Un perfil puede definir también la sintaxis de otros objetos utilizados por la aplicación para la que fue diseñado.

El perfil PKIX, creado por IETF, es uno de los más utilizados, porque facilita el uso de los certificados X.509 en Internet. Los componentes de una PKI, según las especificaciones PKIX [HPF02] son:

- ***Entidad Final(EE):*** Puede ser un cliente de los servicios brindados por la PKI, un sistema de usuario final, un dispositivo, un proceso o cualquier cosa identificada como el sujeto de un certificado de clave pública.
- ***Autoridad de Certificación (CA):*** Es la entidad que expide los certificados de clave pública.
- ***Autoridad de Registro (RA):*** Es una entidad opcional, a la que una CA delega ciertas funciones de gestión, relacionadas con el registro de las entidades finales.
- ***Repositorio:*** Es el método de almacenamiento y recuperación de certificados y CRLs utilizado. Sirve como medio de distribución de los certificados y CRLs a las entidades finales.

2.3.5 Wireless Application Protocol PKI (WPKI)

Dada la reciente convergencia de las comunicaciones móviles con Internet es necesario adaptar los servicios de seguridad a este nuevo entorno, donde la capacidad limitada de los dispositivos y las características del medio son todo un reto para los investigadores. Las amenazas a las que pueden estar expuestas las redes como ataques de denegación de servicio, manipulación de información privada, acceso a los servicios por usuarios no autorizados o compromiso de un nodo, hacen que las operaciones realizadas por los dispositivos móviles requieran seguridad, más aún, cuando hablamos de espacios abiertos, presencia de usuarios desconocidos, usuarios vagando por distintas redes, redes sin gestión centralizada e información (posiblemente sensible) viajando por enlaces inalámbricos. Es también importante proteger los dispositivos móviles porque aunque tengan capacidades limitadas, son personales y por lo tanto, poseen información confidencial. Asimismo, al tener recursos limitados, se debe evitar que sean agotados por usuarios maliciosos mediante ataques de denegación de servicio (DoS –*Denial of Service*).

WPKI [WAP01b] es una optimización de la PKI tradicional para el entorno inalámbrico. Se han optimizado: los protocolos (usando WML [WAP01c] y WMLSCrypt [WAP01d]), el formato de los certificados (certificados WTLS [WAP01e]) y los algoritmos criptográficos y

claves (criptografía de curvas elípticas ECC [Cer00]). El objetivo de WPKI es reutilizar los estándares de PKI y desarrollar nuevos estándares únicamente donde sean necesarios los requerimientos específicos de WAP(*Wireless Application Protocol*) [WAP01a].

El modelo general adoptado actualmente por WPKI es el siguiente [WAP01b]:

- Los certificados WTLS del servidor y la CA raíz almacenados en un dispositivo tendrán el formato de los certificados WTLS definidos en [WAP01e].
- Los certificados del cliente (WTLS y aplicación) y de la CA raíz almacenados en los servidores tendrán el formato de los certificados X.509 definidos en [HPF99].
- Los certificados del cliente (WTLS y aplicación) y de la CA raíz que deban transmitirse vía radio y/o sean almacenados en los dispositivos de los clientes WAP tendrán el formato de los certificados X.509 especificado en [WAP01g].
- El almacenamiento de la URL del certificado, en lugar del certificado completo, es preferible cuando los certificados en formato X.509 vayan a ser transmitidos vía radio.
- El almacenamiento de certificados X.509 en el dispositivo móvil será excepcional, sólo si son proporcionados con el dispositivo, a través de un módulo WIM (*Wireless Identity Module*) [WAP01f], por ejemplo.

WPKI comprende los mismos componentes de PKI: CA, RA, EE, repositorio (DIR); y añade un Portal PKI, que traduce las peticiones hechas por los clientes WAP a las CAs/RAs de la PKI, es decir, interactúa con los dispositivos WAP de la red inalámbrica y las CAs de la red cableada (Figura 2.1).

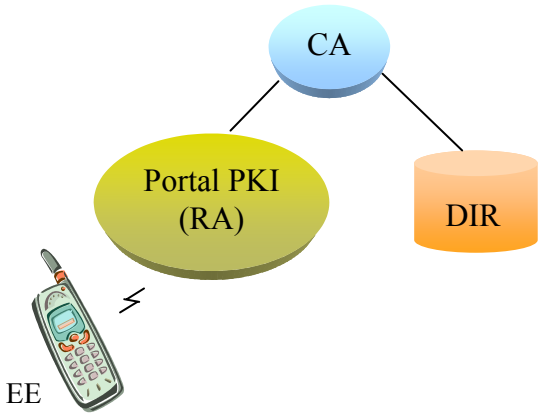

Figura 2.1: Arquitectura WPKI

2.3.6 Estándares PKI

Las actividades de estandarización de los conceptos y técnicas relacionados con PKI pueden clasificarse en diferentes grupos: aquellos que definen el formato de los certificados, donde se incluyen X.509, SPKI, OpenPGP y EDIFACT; aquellos que adaptan los certificados a ambientes y usos específicos (principalmente certificados X.509v3), entre los que se encuentran PKIX, TC68, S/MIME, IPSec, TLS, WAP, XMLdsig, XMLenc, y SOAP; y los que tratan temas relacionados con los repositorios de certificados y CRLs (de nuevo, principalmente, para formatos X.509), que es hacia donde dirigen sus esfuerzos X.500, LDAP y XKMS [AL03].

2.3.6.1 X.509

Esta recomendación [ITU00] define y estandariza un formato de certificados general y flexible. La amplia difusión de este formato muestra su idoneidad para diversos ambientes y su disponibilidad como estándar internacional justo en el momento en que un gran número de proveedores iniciaban la implementación de sus productos.

La utilidad real de X.509 proviene de su poderoso mecanismo de extensiones definido para la versión 3 de los certificados y la versión 2 de las CRLs. Así, el contenido de los certificados y las CRLs puede ser adaptado fácilmente a ambientes específicos restringiendo o no su utilidad en otros ambientes.

Aunque éste es un estándar internacional, X.509 ha continuado evolucionando en algunos aspectos. Los problemas encontrados a nivel operacional han sido (y continúan siendo) tratados en el estándar a través de reportes formales de defectos y procesos de resolución. También se le han incorporado algunas modificaciones para aclarar el texto o para proporcionar detalles adicionales; un ejemplo de esto es la definición y uso de los certificados de atributos para la gestión de privilegios.

2.3.6.2 PKIX

Un grupo de trabajo del área de seguridad del IETF es el de Infraestructura de Clave Pública, X.509, comúnmente conocido como PKIX. Este grupo fue formado a finales de 1995 con el único propósito de adaptar el trabajo hecho por la recomendación X.509 al entorno de Internet, especificando así una IPKI (Internet PKI). PKIX ha tratado cuatro áreas específicas:

- La adaptación de los certificados y las CRLs.
- Los protocolos de gestión de certificados.
- Los protocolos operacionales.
- El marco de políticas de certificados (CP – *Certificate Policy*) y el de declaración de prácticas de certificación (CPS – *Certification Practice Statemen*t).

Entre las especificaciones que ha definido el grupo de trabajo PKIX tenemos: RFC2459 (reemplazada ahora por el RFC3279 y el RFC3280), RFC2510, RFC2511, RFC2527, RFC2559, RFC2560, RFC2585, RFC2587, RFC2797, RFC3029, RFC 3039, RFC3161, y RFC3281.

2.3.6.3 X.500

El directorio X.500 [ITU01] es un repositorio altamente sofisticado que almacena todo tipo de información e incluye características como: protocolos de acceso cliente a directorio, protocolos de comunicación servidor a servidor, duplicación total o parcial de los datos del directorio, encadenamiento de servidores para responder a una petición, y capacidades de filtrado de búsquedas complejas.

La recomendación X.500 define un esquema de particular importancia para PKI porque permite guardar estructuras de datos, como certificados y CRLs, en las entradas de directorio de cada entidad. Además, la posibilidad de enlazar servidores, hace posible que las PKIs desarrolladas de manera independiente puedan unirse cuando sea necesario, lo que permite comunicaciones seguras entre comunidades aisladas.

2.3.6.4 LDAP

LDAP (*Lightweight Directory Access Protocol*) [YHK95] fue originalmente concebido como un subconjunto del protocolo de acceso al directorio X.500 (DAP – *Directory Access Protocol*), simple de describir y fácil de implementar. Luego, este subconjunto de funciones se ha expandido para incorporar las necesidades de otros entornos que han elegido usar LDAP como protocolo de acceso a sus repositorios.

Muchos proveedores a nivel mundial implementan LDAP versión 2 [YHK95]. LDAPv3 [WHK97], con su útil mecanismo de extensiones permite incorporar nuevas capacidades instantáneamente de manera estandarizada, y es recomendado en lugar de LDAPv2.

Como con X.500, se ha especificado un esquema para repositorios compatibles con LDAP [BHR99] que provee un método estandarizado y un lugar para almacenar información de certificados y CRLs a las entidades PKI. Esto realza grandemente la posibilidad de interoperabilidad entre productos PKI de diferentes proveedores en el entorno LDAP.

2.3.6.5 S/MIME

El propósito inicial del grupo de trabajo S/MIME fue incorporar nuevas características de seguridad en MIME (*Multipurpose Internet Mail Extensions*) (RFC 2045 a RFC2049), mientras se mantenía la compatibilidad con productos implementados de acuerdo a la versión previa de las especificaciones. En particular, las especificaciones S/MIMEv3 (RFC2630 y RFC2634) incluyen la habilidad de etiquetar los mensajes de acuerdo a su nivel de seguridad, y la habilidad de solicitar y obtener un recibo firmado como prueba de que se recibió un mensaje enviado previamente. También se pueden usar técnicas de gestión de claves diferentes a RSA (por ejemplo, Diffie-Hellman).

Las especificaciones S/MIMEv3 definen una versión de los certificados X.509 que es compatible con PKIX y a su vez especifica extensiones relevantes para S/MIME.

2.3.6.6 IPSec

El grupo de trabajo IPSec del IETF fue creado para diseñar y estandarizar los conceptos y protocolos de seguridad en IP (*Internet Protocol*). Un componente importante de esta arquitectura es el protocolo de intercambio de claves entre nodos IP, que provee autenticidad, integridad y confidencialidad. Esto es expresado en IKE (*Internet Key Exchange*) [HC98]. IKE proporciona autenticación basada en certificados X.509 en la capa IP y puede ser compatible con PKIX.

2.3.6.7 TLS

La especificación TLS (*Transport Layer Security*) [DA99] es una versión del protocolo SSLv3.0 (*Secure Sockets Layer version* 3.0) encontrada en millones de navegadores cliente y servidores Web alrededor del mundo. TLS crea un canal seguro entre la fuente y el destino en la capa de transporte, proveyendo autenticación basada en certificados, integridad de la información y confidencialidad de los datos.

2.3.6.8 SPKI

El grupo de trabajo SPKI (*Simple Public Key Infrastructure*) del IETF fue creado en 1996 como alternativa al trabajo hecho por PKIX. Los partidarios de SPKI defienden la idea de la clave pública como identidad principal. Las especificaciones SPKI (RFC2692 y RFC2693) discuten los conceptos y filosofía de las IPKIs y proveen un formato de certificados detallado y las reglas de procesamiento requeridas para su implementación. SPKI incluye explícitamente tanto la autorización como la autenticación. El sofisticado formato de los certificados hace posible expresar, de manera general, qué se puede hacer con una clave.

Sin embargo, SPKI no ha ganado un amplio soporte en el entorno corporativo y gubernamental como X.509. Pocos proveedores han incluido certificados SPKI en sus productos.

2.3.6.9 OpenPGP

OpenPGP define un formato de certificados para PGP (*Pretty Good Privacy*), alternativo a los certificados X.509 y los de SPKI., y especifica las reglas de procesamiento requeridas para validar tales certificados. También define los protocolos de empaquetamiento requeridos para construir y procesar mensajes de correo electrónico protegidos con PGP.

La especificación OpenPGP también incluye una discusión sobre el trabajo cooperativo con otros formatos de certificados como los X.509.

A pesar de una base instalada de tamaño significativo, OpenPGP no ha capturado una gran porción de las infraestructuras de seguridad corporativas y gubernamentales. Aún así, parece probable que OpenPGP continuará siendo popular para los usuarios de Internet individuales.

2.3.6.10 WAP

Las especificaciones definidas por el grupo de seguridad WSG (*WAP Security Group*) del WAP Forum incluyen: *Wireless Transport Layer Security Specification* [WAP01e], *WAP Certificate and CRL Profiles Specification* [WAP01g], y *WAP Public Key Infrastructure Definition* [WAP01b]. En el modelo general WPKI, el servidor usa el formato de certificados WTLS, mientras los clientes usan el formato de los certificados X.509 (para interoperar lo mayor posible con las IPKIs existentes). Además, la especificación WTLS permite varios niveles de seguridad, incluyendo un intercambio de claves anónimo para crear canales cifrados, autenticación de servidor usando certificados y autenticación mutua basada en certificados (esto es, de parte del cliente como del servidor).

2.3.6.11 XML

Organismos importantes de estandarización desarrollaron y adoptaron las especificaciones de seguridad XML (*eXtensible Markup Language*), entre ellos W3C (*World Wide Web Consortium*) y OASIS (*Organization for the Advancement of Structured Information Standards*). Dentro de W3C, se especificó la sintaxis de XML con respecto al cifrado (XML *Encryption*) y la firma digital (XML *Signature*), y los protocolos XML de gestión de claves (XML *Key Management Specification*) que le permiten al cliente obtener información de claves a través de un servicio Web.

Dentro de OASIS, el comité técnico de servicios de seguridad desarrolló SAML (*Security Assertion Markup Language*) [HM02], y un marco XML para el intercambio de información de autenticación y autorización. Esta información de seguridad es expresada en forma de afirmaciones sobre sujetos que tienen una identidad en algún dominio seguro. El mecanismo de autenticación utilizado puede basarse en PKI, aunque SAML adopta otras tecnologías de autenticación de manera que puede usarse en muchos entornos.

2.4 Autorización

La autorización está estrechamente ligada con la autenticación. Una vez el usuario ha validado su identidad para acceder a algún recurso, es necesario restringir sus acciones de acuerdo a quién es, qué está tratando de hacer, etc. [SSW05]. La autorización dota al usuario de privilegios para poder efectuar ciertas operaciones con los datos o recursos protegidos.

2.4.1 Modelos de Control de Acceso

Un factor fundamental para determinar el funcionamiento de todo entorno de autorización es la definición del modelo de control de acceso que se va a establecer.

Los mecanismos utilizados para restringir el acceso a los recursos son generalmente de una (o algunas veces la combinación) de dos formas:

- ***Control de Acceso Discrecional (DAC):*** Le deja las decisiones de control de acceso al propietario del recurso, de manera que es él/ella quien decide qué sujetos pueden realizar determinadas acciones sobre los recursos poseídos. Así, un sujeto con permisos de acceso puede otorgarlos (quizás indirectamente) a otro sujeto [NCS88].
- ***Control de Acceso Obligatorio (MAC):*** Este control de acceso se basa en las reglas establecidas por una autoridad central. MAC restringe el acceso a los objetos basándose en la sensibilidad de la información que estos contienen (representada por una etiqueta) y en la autorización formal de los sujetos para acceder a dicha

información [NCS88]. Los objetos son considerados como entidades pasivas que almacenan información, mientras que los sujetos son entidades activas que realizan peticiones de acceso a los objetos.

Existen diversos modelos de control de acceso de tipo DAC y/o MAC. Los más comunes son [YT05]:

- ***Control de Acceso Basado en Identidad (IBAC):*** En este caso, los permisos de acceso a un recurso se asocian directamente al identificador del sujeto (es decir, el nombre del usuario). Por tanto, se garantiza el acceso al recurso sólo cuando existe dicha asociación.

  Con IBAC no se asocian etiquetas de seguridad a los usuarios, por lo que éste es primordialmente un mecanismo de control de acceso discrecional.

  Un ejemplo de IBAC son las Listas de Control de Acceso (ACLs) [Lam71], encontradas comúnmente en sistemas operativos y servicios de seguridad en red. Una ACL contiene los identificadores de los usuarios junto con sus derechos de acceso a un recurso determinado, como leer, escribir, ejecutar, etc. Esta estructura básica de autorización únicamente extiende el concepto de autenticación, ya que si el usuario no puede autenticarse correctamente ante el guardián del recurso, su solicitud de acceso es denegada.

  Entre más usuarios soliciten el acceso a un recurso más identificadores contendrá la ACL, lo que dificulta el manejo de estas listas y las hace una alternativa poco escalable.

  Por otra parte, la decisión de control de acceso no depende de alguna función o característica de la organización a la que pertenece el usuario sino solamente de los identificadores, así que su uso es inapropiado a nivel empresarial.
- ***Control de Acceso Basado en Roles (RBAC):*** Restringe el acceso a los recursos basándose en la función o rol que desempeña el sujeto dentro de la organización. Los permisos para acceder a un recurso son asignados a cada rol, en lugar de asociarlos

directamente al identificador del sujeto. El RBAC es escalable y reduce significativamente la cantidad de información de administración necesaria, ya que los permisos no son asignados constante e individualmente a los usuarios. RBAC es primordialmente un mecanismo de control de acceso discrecional. Generalmente, no tiene en cuenta las características de los recursos (más que sus identificadores) y no obtiene ninguna información relevante de seguridad del entorno.

PMI implementa un modelo de control de acceso basado en roles. Aquí, se asigna un conjunto de privilegios a cada rol, y uno o varios roles a cada usuario Así se desvinculan los privilegios del usuario de su identidad local, lo que permite un control de acceso más flexible y dinámico.

- ***Control de Acceso Basado en Atributos (ABAC):*** En ABAC, los privilegios son establecidos en base a la colección de atributos que posee el usuario y una política que los determina. ABAC es la convergencia natural de los modelos de control de acceso IBAC y RBAC. La representación de las políticas en ABAC es semánticamente más rica y expresiva. Además, posee una mayor granularidad ya que puede basarse en cualquier combinación de atributos de sujeto, de recursos y de entorno.

  PMI utiliza certificados de atributos para asignar un conjunto privilegios a cada usuario. El verificador comprueba en la política de control de acceso si el usuario tiene los privilegios suficientes para acceder al recurso solicitado.
- ***Control de Acceso Basado en Malla (LBAC):*** En este caso, una colección totalmente ordenada de etiquetas de seguridad se combina con un grupo de categorías y forman una *malla*. Con ello se obtiene un conjunto de clases de seguridad que varía desde la más baja a la más alta [San93]. Generalmente, el modelo LBAC es manejable cuando existe un número relativamente pequeño de etiquetas de seguridad y categorías, por lo que es sólo efectivo para ciertos escenarios de seguridad poco granulados y carentes de flexibilidad y escalabilidad. LBAC es un mecanismo de control de acceso obligatorio.

2.4.2 Estándares del Servicio de Autorización

En esta sección se describen las principales tecnologías utilizadas actualmente para la representación de información de autorización, como son los certificados de atributos X.509, SPKI/SDSI y SAML. Estas tecnologías influyen notablemente en el diseño, implementación y puesta en marcha de la solución de autorización empleada, así como en su escalabilidad, extensibilidad e interoperabilidad con otros entornos.

2.4.2.1 Certificados de Atributos X.509

En un principio, la ITU-T implementó el servicio de autorización en los certificados de clave pública adicionándoles la extensión *subjectDirectoryAttributes*. Pero estos certificados fueron pensados para un período de vigencia relativamente largo en comparación con los derechos de acceso o privilegios del usuario que tienden a cambiar frecuentemente, lo que ocasiona una inevitable avalancha de revocaciones, reduce la funcionalidad del sistema y disminuye la calidad del servicio prestado. De allí que en la tercera y cuarta edición de la recomendación X.509 [ITU00], se introduzca el concepto de certificado de atributos y se defina el marco para su aplicación en una PMI.

Los *Certificados de Atributos* (ACs), no son más que estructuras de datos que asocian a una entidad un conjunto de atributos o privilegios.

La sintaxis de un AC guarda cierta similitud con la de los PKCs (Figura 2.2), porque contiene los campos habituales de versión (*version*), número de serie (*serialNumber*), firma (*signature*), emisor (*issuer*), período de validez (*attrCertValidityPeriod*), e incluso los campos opcionales identificador único de emisor (*issuerUniqueID*) y extensiones (*extensions*). Existen, sin embargo, campos nuevos como son el del titular o propietario del certificado (*subject/holder*) y el de atributos (*attributes*), que contiene información sobre los privilegios que tiene el titular, como pertenencia a grupos, identificación de cargos, valores límite de transacciones, horas de realización de ciertas operaciones, etc.

Es conveniente resaltar que, a diferencia de los PKCs, la identificación del usuario no se deja explícita en el certificado de atributo, sino que se utiliza el campo *subject/holder* para enlazar este certificado con el correspondiente PKC del usuario, especificando para ello su

número de serie y emisor (*IssuerSerial*). Igualmente sucede con el campo *issuer*, donde se proporciona información sobre el PKC de la autoridad que expide este certificado. De esta forma, la PKI autentica aquellos usuarios a los que la PMI emite ACs.

Adicionalmente, el campo *subject/holder* puede contener un hash (*ObjectDigestInfo*) de la clave pública o bien del PKC completo. También puede contener algún identificador del usuario (*GeneralNames*), utilizado en el caso en que la PKI coexista con algún otro esquema de autenticación.

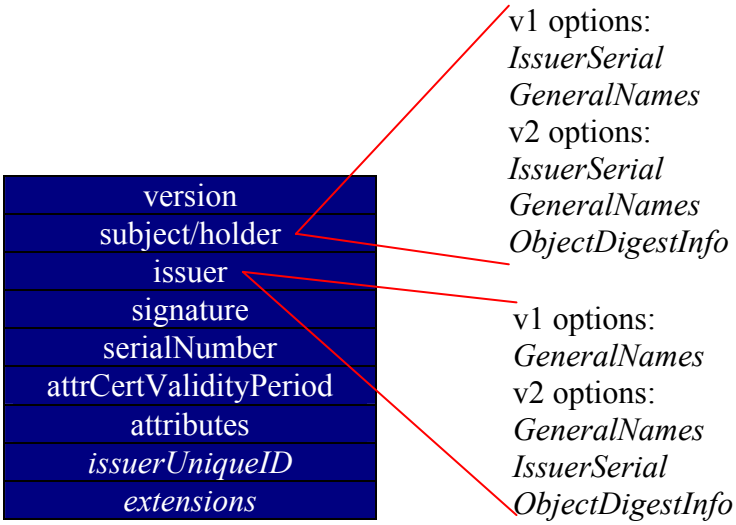

Figura 2.2: Sintaxis de un certificado de atributos

Arquitectura

La Infraestructura de Gestión de Privilegios se define como:

> "*Infraestructura capaz de soportar la gestión de privilegios de un servicio de autorización completo, relacionado con una infraestructura de clave pública*".

En definitiva, PMI es el conjunto de hardware, software, gente, políticas y procedimientos necesario para crear, manejar, guardar, distribuir y revocar certificados de atributos.

En PMI, los certificados de atributos son firmados digitalmente por una TTP llamada *Autoridad de Atributos* (AA), que es la encargada de asignar los privilegios y que, según la ITU, no tiene porque ser la misma que emite los certificados de clave pública. Los titulares o propietarios de los certificados, se denominan *Asertores de Privilegios,* una vez afirman sus derechos ante un verificador de privilegios para obtener el acceso a algún recurso. Es el *Verificador de Privilegios* quien debe tomar la decisión de otorgarles o no dicho acceso, con base en los privilegios del usuario, las políticas vigentes, las variables del entorno actuales y las características particulares del objeto al que se desea acceder. Para que el verificador de privilegios pueda desempeñar su función debe recurrir a los servicios de la PKI y extraer información de los repositorios de certificados.

PMI puede usar varios modelos en función de las características de cada aplicación. Así, hay un modelo general, y sobre éste se definen 3 modelos específicos: modelo de control de acceso, de roles y de delegación.

- ***Modelo General:*** Consta de tres entidades: objeto, asertor de privilegios y verificador de privilegios. El objeto es el recurso que se pretende proteger. El asertor de privilegios es la entidad a la que se han asignado los privilegios, mientras que el verificador de privilegios es quien toma la determinación de si son o no suficientes los privilegios afirmados para realizar una determinada operación sobre el objeto (Figura 2.3)

Figura 2.3: Modelo general

- ***Modelo de Control de Acceso:*** Aquí, el verificador de privilegios examina todas las peticiones que realiza el asertor y sólo tienen acceso a los métodos del objeto

(acciones sobre el objeto) aquellas que están adecuadamente autorizadas, según la política establecida y las variables del entorno (Figura 2.4).

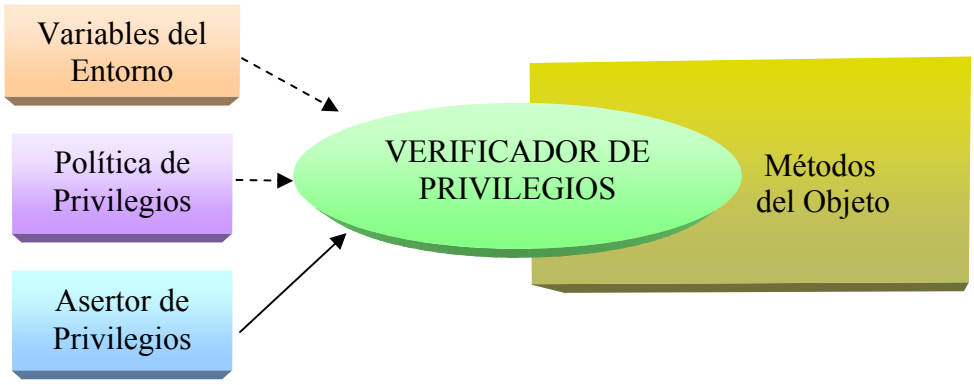

Figura 2.4: Modelo de control de acceso

- ***Modelo de Delegación:*** Se utiliza en aquellos escenarios en que no sólo es necesario asignar privilegios, sino también proporcionar mecanismos para que las entidades puedan delegar los que les han sido otorgados. La *Fuente de Autoridad* (SOA – *Source Of Authority*) es un tipo específico de AA y la .responsable inicial de la asignación de estos privilegios. También, autoriza a otras entidades para que se conviertan en AAs y puedan a su vez delegar todos o parte de los privilegios que se les otorgan a otras AAs o directamente a entidades finales. Por supuesto, las autoridades de atributos no pueden delegar más privilegios que los que se les han otorgado y ésta delegación debe hacerse de acuerdo a las políticas establecidas por la SOA, especificadas en el propio certificado. Con ello se forman caminos de delegación que constan de una serie de ACs enlazados por los nombres de sus emisores y titulares. El verificador de privilegios deberá validar dicho trayecto comprobando que cada AA tiene los privilegios y la autorización suficiente para la delegación (Figura 2.5).

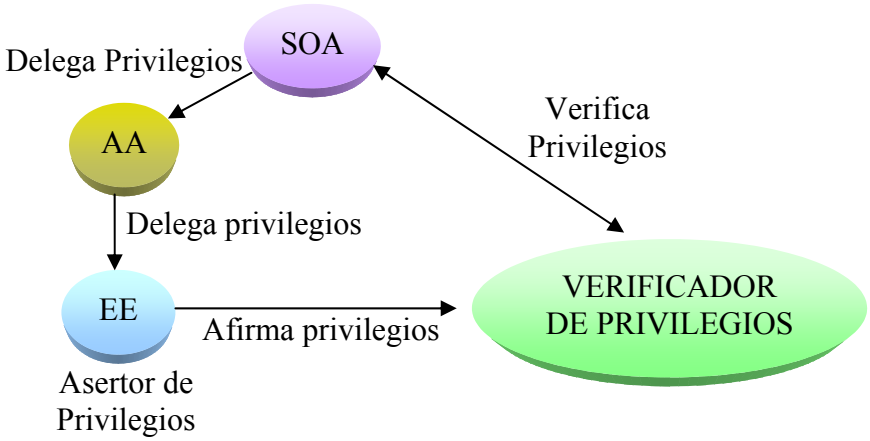

Figura 2.5: Modelo de delegación

- ***Modelo de Roles:*** Se basa en el uso de roles para asignar privilegios a los usuarios de forma indirecta. Utiliza dos tipos de certificados: los certificados de asignación de roles (mediante los cuales se otorga uno o varios roles a cada usuario) y los certificados de especificación de roles (en los que se le asigna un conjunto de privilegios a cada rol). El verificador de privilegios debe entonces descubrir los privilegios vinculados con cada rol. El camino de delegación para un certificado de asignación de roles puede implicar diferentes AAs y ser independiente de la AA que expidió el certificado de especificación de roles (Figura 2.6).

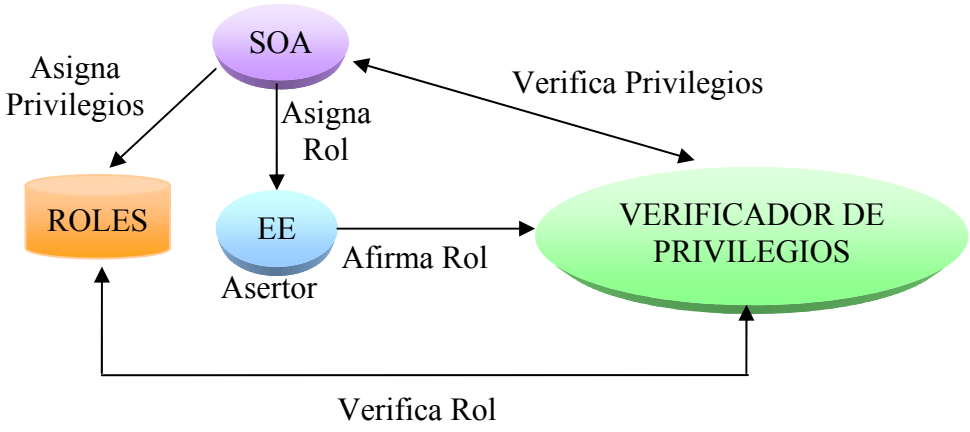

Figura 2.6: Modelo de roles

2.4.2.2 SPKI/SDSI

En [RL96], Rivest y Lampson propusieron una nueva infraestructura de clave pública denominada *Simple Distributed Security Infrastructure* (SDSI), en la que se definen nombres locales para las entidades, en contraposición al entorno PKI donde la definición de nombre se realiza de modo global. Casi al mismo tiempo, Carl Ellison [Ell99] desarrolló *Simple Public Key Infrastructure* (SPKI), centrada en como gestionar la delegación en el servicio de autorización. En 1999, ambas propuestas se unen con el nombre de SPKI/SDSI [EFL99].

SPKI/SDSI puede verse como un lenguaje que ayuda a gestionar la confianza en sistemas descentralizados donde las decisiones de acceso se basan en sentencias de políticas definidas por las entidades. Estas sentencias de políticas toman la forma de certificados. SPKI define tres tipos de certificados: certificados de identidad, que vinculan un nombre de usuario con su clave; certificados de atributo, que vinculan la autorización del usuario con su identidad; y certificados de autorización, que vinculan la autorización del usuario con su clave. Estos certificados son firmados digitalmente por su emisor y tienen un período de validez limitado.

SPKI soporta el concepto de delegación, por lo que mediante los certificados de autorización o de atributo se puede autorizar la propagación de toda o parte de la autoridad que recibe el emisor del certificado.

SPKI/SDSI despertó gran interés a finales de los noventa y principios del dos mil, pero el auge de los sistemas basados en XML y los servicios definidos a su alrededor, ha hecho que esta tecnología pase a un segundo plano.

2.4.2.3 SAML (Security Assertion Markup Language)

Desarrollado por OASIS, es un lenguaje estándar basado en XML para intercambiar información de autenticación y autorización entre dominios de seguridad, es decir, entre un proveedor de identidad y un proveedor de servicios.

En SAML, la información es expresada en forma de aserciones sobre los sujetos, donde un sujeto es una entidad (ser humano u ordenador) identificada en algún dominio seguro. Una aserción puede incluir datos acerca de un acto de autenticación realizado sobre una entidad (aserción de autenticación), información de atributos sobre la entidad (aserción de atributos) o

información de decisiones de autorización sobre un recurso (aserción de decisión de autorización). Estas aserciones son emitidas por autoridades SAML, llamadas autoridades de autenticación, autoridades de atributos y puntos de decisión de políticas, respectivamente. SAML define un protocolo mediante el cual los clientes pueden enviar aserciones a las autoridades SAML y obtener respuesta a ellas.

La aserción de atributos puede ser vista como un directo competidor de los certificados de atributos, ya que es el mecanismo utilizado para transportar los atributos de los usuarios. Uno de los inconvenientes que limita el uso de la aserción de atributos es que está definida en XML, y tal como se demuestra en [MC04], el tratamiento de los atributos en XML es menos eficiente, tanto en tamaño como en tiempo de procesamiento, que en formato ASN.1.

Entre las principales ventajas de SAML cabe destacar:

- Independiza la seguridad de la lógica de aplicación.
- No requiere que la información de usuario sea mantenida y sincronizada entre directorios.
- Permite utilizar mecanismos SSO que le posibilitan a los usuarios identificarse ante un proveedor de información y tener acceso a distintos proveedores de servicio sin necesidad de autenticaciones adicionales.
- Los mecanismos SSO permiten reducir el coste de mantenimiento de la información
- La responsabilidad del mantenimiento de la información relacionada con la identidad y la autorización de los usuarios es transferida del proveedor de servicios al gestor de identidad.

2.4.3 Gestión de Autorización en Entornos Web

Una federación puede definirse en forma genérica como el acto de establecer una relación de confianza entre dos entidades. En forma más detallada, puede entenderse como una delegación del control de los servicios de seguridad a los proveedores de identidad.

En esta sección analizamos las soluciones de federación más destacadas en el ámbito educacional (Shibboleth, PAPI) y empresarial (*Liberty Alliance*).

2.4.3.1 Shibboleth

Shibboleth [SC05] es un proyecto de Internet2/MACE, que proporciona acceso a información vía web empleando una infraestructura de autorización basada en SAML. El propósito de esta propuesta es determinar si un usuario, usando su navegador web, tiene los permisos requeridos para acceder a un recurso, de acuerdo a los atributos que ese usuario tiene previamente asignados en su dominio de origen. Por consiguiente, Shibboleth es un sistema para la transmisión segura de los atributos de usuario del dominio de origen al dominio del proveedor del recurso. Este sistema preserva la privacidad del usuario, ya que le asocia a cada uno un manejador, donde se almacena la información de seguridad sin exponer la identidad del usuario. En el intercambio de información realizado por Shibboleth intervienen dos componentes: la autoridad de atributos (AA) en el dominio del usuario, y el solicitante de atributos Shibboleth (SHAR – *Shibboleth Attribute Requester*) en el lado del recurso. Una vez el SHAR conoce el manejador del usuario que solicita el acceso a un recurso, envía un mensaje AQM (*Attribute Query Message*) a la AA del dominio de origen para obtener los atributos del usuario, al que la AA responde con un mensaje ARM (*Attribute Response Message*).

2.4.3.2 PAPI

PAPI (*Point of Access to Providers of Information*) [CL01] ha sido desarrollado por RedIris, para gestionar el control de acceso en entornos web entre proveedores de recursos o información y usuarios finales. Este sistema permite que el usuario sea autentique en su propio dominio mediante credenciales basadas en nombre de usuario y contraseña o certificados de identidad. La idea principal es mantener la autenticación como una cuestión local a las organizaciones donde pertenece el usuario, mientras los proveedores de información tienen pleno control de los recursos que ofrecen.

PAPI consta de dos componentes: el servidor de autenticación (AS – *Authentication Server*) y el punto de acceso (PoA – *Point of Access*). El AS ofrece a los usuarios un punto único de autenticación para obtener el conjunto de credenciales requeridas para acceder a los servicios de un modo transparente. Este servidor también puede actuar como AA

proporcionando los atributos de usuario a los puntos de acceso que los soliciten. Por otra parte, el PoA gestiona el control de acceso y autorización en un conjunto de servicios web. Los puntos de acceso pueden estructurarse de modo jerárquico para permitir una mejor relación con los servicios.

En una arquitectura basada en PAPI la autenticación se realiza en el dominio local del usuario. Una vez autenticado, puede solicitar su acceso a un recurso dirigiéndose al PoA que lo controla. El AS tras la autenticación correcta del usuario, crea una sentencia de autenticación que firma digitalmente. El PoA del recurso destino recibe esta sentencia de autenticación, comprueba su firma, si el AS está dentro de sus entidades de confianza, y decide si concede o no el acceso al usuario. De este modo, el proceso de autorización es local al dominio que gestiona los recursos a proteger.

Como ventajas de PAPI cabe destacar su simplicidad, gracias al uso de cookies HTTP para transportar la información de autenticación hacia los puntos de acceso al servicio.

2.4.3.3 Liberty Alliance

El proyecto *Liberty Alliance* [BK03] pretende crear una solución abierta y federada de SSO que sea utilizada por las aplicaciones comerciales mediante cualquier dispositivo conectado a Internet. Actualmente, el proyecto cuenta con más de 150 organizaciones miembro.

El proyecto *Liberty Alliance* ofrece un entorno de gestión de identidades basado en SAML para servicios web, de modo parecido a Shibboleth. La idea principal es la de ofrecer un marco para el establecimiento de federaciones de identidades en Internet, compuestas por los proveedores de esas identidades y proveedores de servicios, que pueda gestionar de modo eficiente y seguro las distintas identidades de cada usuario.

Los principales objetivos de *Liberty* son:

- Ofrecer privacidad a las identidades de los usuarios a la hora de acceder a los servicios de Internet.
- Posibilitar la gestión de negocios sin intervención de terceras partes.
- Ofrecer un estándar de SSO para entornos web.

- Crear una infraestructura de identidad de red.

Liberty se basa en la existencia de proveedores de identidad, proveedores de servicios, usuarios finales y en el uso de la redirección web para la comunicación entre ellos.

Cuando el usuario accede a un servicio web protegido por un proveedor de servicios *Liberty*, es redirigido, mediante HTTP, hacia el proveedor de identidad seleccionado por el usuario. El usuario se autentica ante este proveedor de identidad usando, por ejemplo, un certificado de identidad o nombre de usuario y contraseña, con lo que se generan las credenciales de autenticación. El proveedor de identidad vuelve a dirigir al usuario hacia el proveedor de servicios, con una referencia a estas credenciales. El proveedor del servicio recibe la redirección, obtiene la referencia y consulta al proveedor de identidad para corroborar la validez de las credenciales.

Aunque tanto *Liberty Alliance* como Shibboleth se basan en el uso de tecnologías SAML y SOAP, actualmente no son compatibles entre sí. Entre sus principales diferencias están: los usuarios Shibboleth no requieren tener cuenta en el destino donde se encuentra el recurso; en Shibboleth los dominios se intercambian atributos de usuario para la toma de decisiones de autorización, mientras en *Liberty* se intercambian un identificador de usuario.

2.5 Validación de Certificados

La validación es parte fundamental de PKI y PMI, porque permite corroborar la integridad, validez y origen de los certificados. A continuación se describe como se lleva a cabo este proceso en las arquitecturas de autenticación y autorización mencionadas.

2.5.1 Validación de Caminos de Certificación

La verificación de firma de un certificado permite corroborar la integridad y autenticidad de su contenido. Para ello, el verificador necesita conocer la clave pública de la autoridad que expidió dicho certificado. Ya que los usuarios no suelen confiar en todas las CAs de la PKI, debe existir cierta relación de confianza entre ellas para que estos puedan comprobar la firma

de los certificados expedidos por las autoridades en que no confían. Estas relaciones de confianza le permitirán al verificador construir cadenas de certificados, donde el sujeto de uno es el emisor del siguiente. Así, un *Camino de Certificación* [HPF02] es una cadena de PKCs, que le permite a un usuario determinado obtener la clave pública de otro.

Para poder confiar en la clave obtenida, el usuario debe verificar la firma y el estado de validez de cada certificado en la cadena.

El camino inicia en la base de confianza (*trust anchor*) del verificador y termina en la clave pública de CA requerida para validar el certificado del otro usuario.

La base de confianza es la CA en la que el usuario confía bajo cualquier circunstancia. Cada usuario recibe la clave pública de su base de confianza en el momento en que se registra en la PKI.

La validación de un camino de certificación involucra los siguientes pasos:

- ***Construcción del Camino de Certificación:*** Es establecer un camino confiable entre el verificador y la entidad objetivo, a través de las CAs de la PKI, basándose en la relación de confianza que existe entre ellas.
- ***Recuperación de Certificados:*** Es recuperar los certificados que hacen parte del camino de certificación de los repositorios donde se encuentran almacenados.
- ***Verificación de Firmas Digitale*s:** Es comprobar la firma digital de cada certificado recuperado.
- ***Verificación de Validez de los Certificados:*** Es confirmar si el certificado recuperado ha expirado o ha sido revocado. La expiración se comprueba con el período de validez del certificado, y la verificación del estado de revocación depende del mecanismo de revocación utilizado.

2.5.1.1 Modelos de Confianza

Los modelos de confianza o arquitecturas de certificación proporcionan un marco tecnológico para la creación y gestión de relaciones de confianza entre las diferentes entidades de una PKI. La confianza siempre involucra una relación entre dos partes y un conjunto de

expectativas formadas sobre dicha relación. Así, si se conoce bien la otra parte y se tienen experiencias que soporten esas expectativas, el nivel de confianza establecido será alto.

Un *dominio de confianza* dentro de una organización puede definirse como un grupo de sistemas bajo el control de un conjunto de políticas comunes. Cuando se construye una PKI, el establecimiento de dominios de confianza y sus límites es muy importante.

El dominio de confianza o de certificación de una CA define el área organizacional o geográfica dentro de la cual la CA es considerada confiable. Todos los usuarios de la PKI dentro del dominio de certificación de una CA, la consideran su base de confianza.

Cuando dos usuarios que pertenecen al mismo dominio de certificación quieren comunicarse, es fácil para uno obtener la clave pública del otro, ya que conocen la clave pública de la CA que expidió sus certificados. Pero cuando hacen parte de dominios de certificación diferentes, esto sólo es posible si existe una cadena ininterrumpida de puntos de confianza entre ellos, lo que supone la intervención de varias CAs y por tanto la necesidad de un acuerdo o correspondencia entre sus políticas. Para ello, las CAs emplean la certificación cruzada, que les permite a los usuarios construir caminos de certificación de un punto a otro.

El término C*ertificación Cruzada* se refiere al establecimiento de la relación de confianza entre dos autoridades de certificación a través de un certificado firmado por una CA y que contiene la clave pública de otra CA, denominado C*ertificado Cruzado* [ITU00].

Los modelos de confianza describen como son construidas estas relaciones y las reglas necesarias para encontrar y recorrer los caminos de certificación.

Las relaciones de confianza pueden ser unidireccionales o bidireccionales. En muchos casos son bidireccionales ya que cuando se decide confiar en alguien, esa confianza suele ser recíproca.

Según las relaciones de confianza que se establecen se forman distintos modelos, los más conocidos son [Lin00], [PH00], [Per99]:

- ***PKI de una Única CA:*** En este modelo una CA provee los servicios PKI a todos los usuarios (Figura 2.7), es decir, todos los usuarios de la PKI confían en la única CA de

la arquitectura. Así, cada camino de certificación comienza con la clave pública de dicha CA.

Esta configuración es la más simple de implementar, pero si la clave pública de la CA cambia, toda la arquitectura debe ser reconfigurada. Además, no es una solución escalable pues no es adecuada para comunidades de usuarios grandes y heterogéneas.

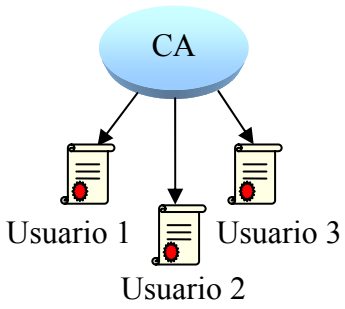

Figura 2.7: PKI de una única CA

- ***PKI Jerárquica:*** Es el modelo más común. En esta configuración, todos los usuarios confían en la misma CA raíz (RCA). Es decir, todos los usuarios de una PKI jerárquica inician sus caminos de certificación con la clave pública de la RCA. En general, la CA raíz no expide certificados a usuarios sino únicamente a CAs subordinadas. Cada CA subordinada puede expedir certificados a los usuarios o a otro nivel de CAs subordinadas, si las políticas de certificación lo permiten (Figura 2.8). En una PKI jerárquica, la relación de confianza es unidireccional, es decir, las CAs subordinadas no expiden certificados a sus CAs superiores.

  Las PKIs jerárquicas son escalables. Los caminos de certificación son fáciles de construir por ser unidireccionales, y el camino más largo es igual a la altura de la jerarquía más uno: un certificado por cada CA subordinada y el certificado de la entidad objetivo, es decir, el certificado de la RCA no se incluye en el camino. Además, los usuarios de una jerarquía saben implícitamente para qué aplicaciones puede ser usado un certificado, de acuerdo con la posición de la CA dentro de la jerarquía.

  Los problemas del modelo jerárquico se deben a la confianza en un solo punto. El compromiso de la clave privada de la RCA ocasiona el compromiso de toda la

arquitectura. Además, la transición de un conjunto de CAs aisladas a una PKI jerárquica puede ser logísticamente impracticable debido a que todos los usuarios deben ajustar sus puntos de confianza.

Este modelo fue especificado para el estándar PEM (*Privacy-Enhanced Electronic Mail*) [Ken93], que no tubo éxito al querer ser la CA universal.

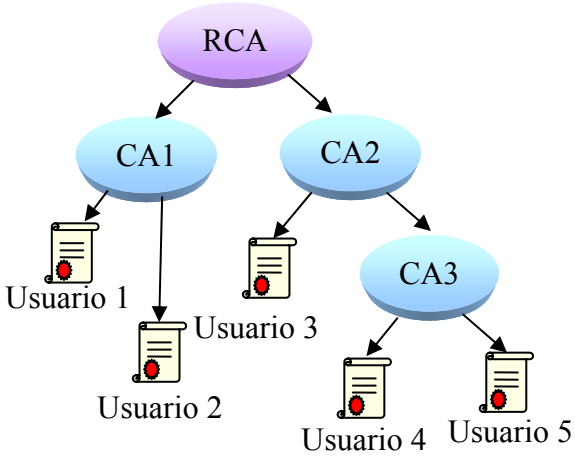

Figura 2.8: PKI jerárquica

- ***PKI en Malla:*** Es también conocida como arquitectura de certificación cruzada. Aquí, la base de confianza de un usuario es su CA local y todas las CAs pueden ser puntos de confianza porque son autónomas. La autonomía se refiere al hecho de que la CA no confía en una CA superior. Una CA autónoma puede realizar certificación cruzada con otras CAs autónomas. Así, una pareja de certificados define la relación de confianza bidireccional entre ellas (Figura 2.9). Sin embargo, esta relación de confianza puede no ser incondicional. Si una CA quiere limitar su confianza, debe especificar estas limitaciones en los certificados que expide a las otras CAs. Toda validación de certificados, de un usuario dentro de una CA autónoma, inicia con el certificado auto-firmado de dicha CA.

  Las PKIs en malla pueden incorporar fácilmente una nueva comunidad de usuarios y aunque el coste de gestión es alto, no hay un único punto de fallo ya que cuenta con

múltiples entidades de confianza. También, pueden existir múltiples caminos entre dos usuarios.

Una PKI en malla puede construirse fácilmente a partir de un conjunto aislado de CAs porque los usuarios no necesitan modificar sus puntos de confianza. Por tanto, este modelo sirve para representar los cambios dinámicos de las estructuras organizacionales o entornos donde las entidades que se comunican no están jerárquicamente relacionadas entre ellas.

La desventaja de esta arquitectura es que el número de relaciones de confianza es directamente proporcional al número de CAs (#CA), es decir, el número de relaciones se incrementará a razón de #CA*(#CA-1), lo que trae problemas de escalabilidad. Además, se requieren certificados de mayor tamaño pues los usuarios deben determinar en qué aplicaciones se puede usar un certificado basándose en su contenido. Por tanto, los certificados tienen más extensiones y el proceso de validación es más complejo.

La longitud máxima de un camino de certificación en una PKI en malla es igual al número de CAs en la PKI.

Aún así, el mayor obstáculo al que debe enfrentarse esta arquitectura es que las aplicaciones que soportan PKI no suelen procesar caminos de certificación con certificados cruzados. Sólo algunas librerías de programación permiten a los desarrolladores de las aplicaciones construir el soporte para certificados cruzados.

Un caso particular de las PKIs en malla, es aquel en el que los propios usuarios son las CAs de la arquitectura, lo que podemos llamar PKI peer-to-peer. Este tipo de modelo de confianza se utiliza en PGP, donde los usuarios se expiden certificados mutuamente y forman la llamada *web of trust*.

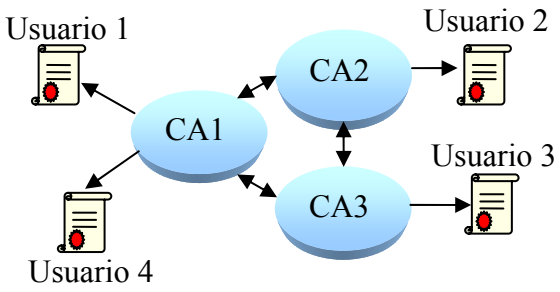

Figura 2.9: PKI en malla

- ***PKI con CA Puente:*** William Polk y Nelson E. Hastings [PH00] emplean una CA puente (BCA – *Bridge Certification Authority*) para establecer la relación de confianza igual a igual entre las diferentes comunidades de usuarios, actuando a manera de hub (Figura 2.10).

  Los usuarios conocen su camino a la BCA y sólo deben hallar el camino de la BCA al certificado de usuario que requieren.

  En este caso, construir los caminos de certificación es más sencillo que en una PKI en malla aunque resulta ser más complicado que en una PKI jerárquica.

  Para implementar una BCA se requiere de información adicional que establezca la relación de confianza entre las CAs de la PKI, lo que quiere decir que los certificados son más complejos y los usuarios deben estar preparados para procesar y usar esta información adicional durante la validación de los caminos de certificación.

  Otro inconveniente al que se enfrenta BCA es que no han sido establecidos aún los mecanismos de distribución y obtención de información de estado de los certificados de una manera útil para los usuarios y sus aplicaciones. En una PKI eficaz, los usuarios deben ser capaces de obtener fácilmente los certificados de las demás entidades de la arquitectura, y la información de estado de validez de estos certificados, a través de un mecanismo de distribución apropiado.

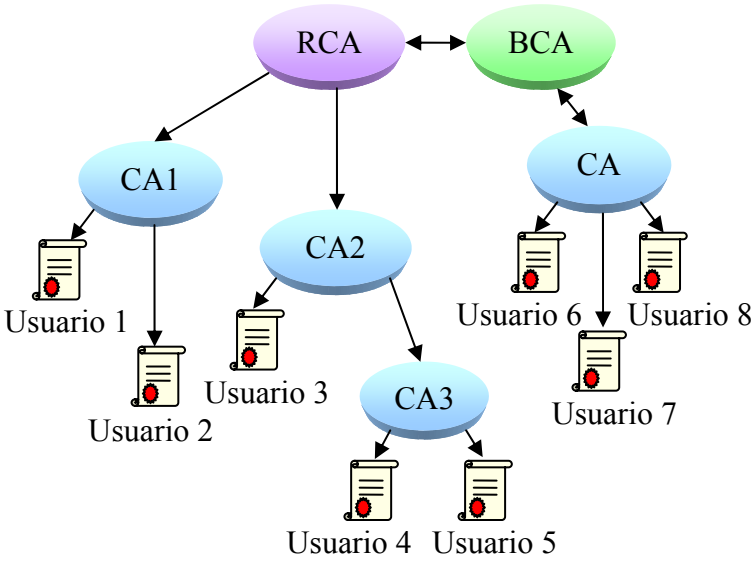

Figura 2.10: PKI con CA puente

- ***PKI Híbrida:*** La naturaleza dinámica de las relaciones de negocios, Internet, etc., hace deseable que los modelos de confianza no sean estáticos y estrechamente limitados. Como su nombre lo indica, el modelo híbrido permite mezclar las arquitecturas de certificación mencionadas anteriormente, por ejemplo, conectar un modelo jerárquico con una PKI de una única CA utilizando certificación cruzada entre sus CAs (Figura 2.11)

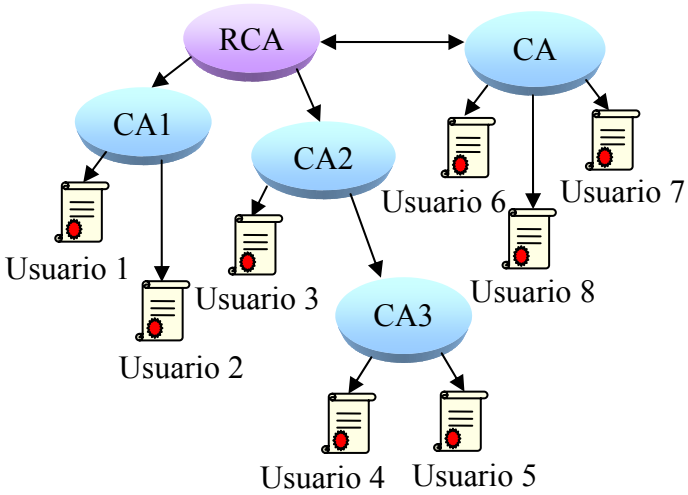

Figura 2.11: PKI híbrida

En definitiva, la relación entre las diferentes comunidades de usuarios determinará qué modelo de confianza es el más apropiado.

2.5.1.2 Métodos de Construcción de Caminos

Para confiar en un PKC, el verificador debe encontrar un camino de certificación desde dicho certificado hasta cualquiera de sus CAs de confianza. Los estándares X.509 [ITU00] y PKIX [HPF02] no especifican ningún método para construir estos caminos. Según Elley et al. [EAH01] existen dos formas básicas de establecerlos, dependiendo de donde se guarden los certificados:

- ***Hacia Adelante (Forward Direction):*** Cuando el camino es construido desde la entidad objetivo hasta la base de confianza. Este método se utiliza cuando el usuario almacena sus propios certificados y se los proporciona al verificador, insertándolos en el mensaje que le envía (modelo *Push*).
- ***Hacia Atrás (Reverse Direction):*** Cuando el camino se construye desde la base de confianza hasta la entidad objetivo. Este método se utiliza cuando es el verificador quien debe recuperar los certificados del usuario, puesto que el mensaje que recibe no los contiene (modelo *Pull*).

Cuando el modelo de confianza es jerárquico, la construcción hacia adelante suele ser la más apropiada, puesto que cada entidad posee un certificado expedido por su CA superior, y sólo existe un camino entre dos usuarios. Para los modelos con relaciones de confianza bidireccionales, sin embargo, la construcción hacia atrás resulta ser la más adecuada, ya que permite rechazar más rápidamente los certificados que no son útiles y se procesan más eficientemente las extensiones involucradas en la construcción del camino.

2.5.1.3 Recuperación de Certificados

Los certificados y la información de revocación deben estar disponibles para los usuarios en algún sitio. El método común para su distribución, en el entorno empresarial, es la publicación, que consiste en colocar la información en un sitio ampliamente conocido, disponible públicamente y fácil de acceder. La publicación es particularmente atractiva para grandes comunidades, donde los usuarios no suelen conocerse entre sí. Comúnmente, esta información se almacena en un repositorio, que es un término genérico usado para denotar cualquier base de datos lógicamente centralizada capaz de guardar información y difundirla cuando es requerido [MOV97].

En el contexto empresarial, los repositorios son típicamente servidores remotos a los que se accede mediante LDAP (*Lightweight Directory Access Protocol*) versión 2 [YHK95]o versión 3 [WHK97].

Aunque LDAPv2 fue el pilar principal de las primeras PKI desarrolladas a nivel empresarial, se considera que es deficiente en las siguientes áreas:

- El mecanismo de autenticación obligatorio entre el cliente y el repositorio se basa en un identificador y una contraseña transmitidos en claro.
- No existe ningún esquema de control de acceso estándar.
- No existe ningún mecanismo estándar para la replicación de datos entre los repositorios LDAP (de hecho, no existe soporte para las comunicaciones servidor a servidor).
- Los filtros de búsqueda se consideran inadecuados.
- No posee la capacidad de realizar operaciones de acuerdo firmadas

Los defectos de LDAPv2 son reconocidos ampliamente, y se han introducido nuevas características con el advenimiento de LDAPv3. Las especificaciones principales de LDAPv3 incluyen los RFCs 2251-2256 y 2829-2830. LDAPv3 soporta mecanismos de autenticación fuerte y se recomienda en lugar de LDAPv2.

Aunque el mecanismo de acceso a los repositorios es LDAP, los repositorios en sí se basan frecuentemente en el modelo de información y los protocolos definidos en la serie de recomendaciones X.500 [ITU01]. Sin embargo, el término repositorio puede aplicarse a una base de datos u otra forma de almacenamiento y distribución de información.

Algunos ejemplos que caen bajo esta definición de repositorio incluyen los siguientes:

- Servidores LDAP
- X.500 Directory System Agents (DSAs)
- Responders OCSP (aunque OCSP como se describe en el RFC 2560 se limita sólo a información de estado de revocación)
- Sistema de nombres de dominio (DNS) (que soporta certificados e información de revocación de acuerdo con el RFC2538)
- Servidores Web (que pueden contener certificados e información de revocación de acuerdo con el RFC2585, recuperables usando el protocolo de transferencia de hipertexto o HTTP).
- Protocolo de transferencia de archivos (FTP) basado en servidores (que puede contener certificados e información de revocación de acuerdo con el RFC2585).
- Bases de datos corporativas (que pueden contener certificados e información de revocación y que poseen prácticas bien definidas de gestión y acceso).

Por tanto, los clientes pueden recuperar información de estos repositorios usando diferentes protocolos de acceso (aunque LDAP es el más utilizado en las PKI empresariales).

La ubicación de un repositorio dado (o un grupo de repositorios) puede ser comunicada al cliente de varias maneras. Por ejemplo, un archivo de configuración del cliente local puede inicializarse con las direcciones IP o los nombres DNS de los servidores LDAP primarios y opcionalmente secundarios, usados por ese cliente.

Las extensiones de los certificados pueden usarse también para indicar la ubicación donde reside la información o servicio deseado.

Uno de los problemas al que debe enfrentarse WPKI, debido a la capacidad limitada de los terminales móviles, es el almacenamiento y gestión de los certificados X.509 que tenga el cliente. Un certificado X.509 puede tener un tamaño considerable (unos 2KB), por lo que la capacidad de almacenamiento que le proporciona al dispositivo un módulo SIM ó WIM [WAP01f] puede ser insuficiente. Por otra parte, el hecho de que el cliente tenga que enviar el certificado completo cada vez que se vaya a autenticar consume mucho ancho de banda. Como solución se ha propuesto el uso de certificados X.509 de tamaño limitado (máximo 700 bytes) [WAP01g], para que puedan ser transmitidos a través de la interfaz radio o almacenados por el dispositivo. Otra alternativa es que el terminal cuente con una dirección URL que apunte al sitio donde está guardado su certificado de manera que no tenga que enviar el certificado completo. Sin embargo, se pueden presentar ataques de negación del servicio si un usuario envía voluntariamente URLs que apunten a certificados que no existen o son falsos.

2.5.1.4 Verificación de Firma de los Certificados

Las CAs se encargan de calcular la firma digital de los certificados que expiden. Para ello, la autoridad obtiene el valor de hash del contenido del certificado y luego realiza una operación de cifrado sobre dicho hash, utilizando su clave privada. Así, para validar la firma de un certificado, el verificador debe conocer la clave pública de la autoridad que emitió dicho certificado y llevar a cabo los siguientes pasos:

1. Descifrar, con la clave pública de la CA correspondiente, la parte firmada del certificado.
2. Calcular un hash sobre el contenido del certificado.
3. Comparar los resultados obtenidos en 1 y 2. Si coinciden, la firma es válida y el verificador puede confiar en el contenido del certificado.

Ya que un certificado contiene la clave pública de su propietario, el verificador puede utilizar esta clave pública para validar la firma de todos los documentos firmados por dicho propietario. De esta manera, a partir de la clave pública de su autoridad de confianza, el verificador puede crear cadenas de certificados y obtener la clave pública de otras entidades.

En WPKI, los algoritmos ECC son usados para mejorar el rendimiento de los terminales móviles. Sin embargo, aunque ECDSA (Elliptic Curve Digital Signature Algorithm) [ANS99] es más rápido para realizar operaciones de firma digital, RSA [RSA78] es más eficiente para realizar operaciones de verificación de firma [TG04]. Por tanto, la elección de uno u otro dependerá de la aplicación que se esté usando.

2.5.1.5 Revocación de Certificados

Revocar es anular el vínculo de una clave pública o privilegio y una entidad, antes de la expiración del certificado que establece dicho vínculo. Un certificado puede ser revocado por la pérdida o compromiso de la clave privada asociada, por un cambio en los derechos de acceso del propietario, etc. Los mecanismos de revocación estándar son CRL [ITU00] y OCSP [MAM99]. Existen también otros métodos alternativos de revocación como CRS, CRT, AD, etc.

- ***CRL (Certificate Revocation List).*** Las CRLs fueron introducidas en 1988 por ITU-T en la recomendación X.509 [ITU00]. Una CRL es una lista, firmada digitalmente por una TTP, que contiene los números seriales de los certificados revocados junto con su fecha y razón de revocación. Estas listas son actualizadas periódicamente y publicadas en repositorios no confiables.

  Para conocer el estado de revocación de un certificado, el verificador debe recuperar la CRL donde se encuentra la información de revocación de dicho certificado y verificar su firma. Luego, debe buscar el número serial del certificado dentro de la lista. Si lo encuentra, el certificado ha sido revocado.
- ***OCSP (Online Certificate Status Protocol).*** Fue adoptado por IETF en 1999 [MAM99]. Sirve para determinar la validez de un certificado en línea.

Para conocer el estado de revocación de uno o más certificados, el verificador envía su solicitud a una entidad confiable llamada responder OCSP. Esta solicitud contiene: la versión del protocolo OCSP utilizado, el tipo de servicio requerido y uno o más identificadores de certificados. Un identificador de certificado contiene a su vez el hash sobre el DN (Distinguished Name) [Kil95] del emisor del certificado, el hash sobre la clave pública del mismo emisor y el número de serie del certificado. El responder OCSP le devuelve al verificador el estado de revocación de estos certificados, junto con sus respectivos identificadores y el intervalo de validez de dicha respuesta. Esta respuesta va firmada digitalmente por el responder OCSP. El estado *good* significa que el certificado no ha sido revocado, pero puede no haber sido expedido todavía o que la respuesta se expidió fuera de su período de validez. El estado *revoked* significa que el certificado se ha revocado y el estado *unknown* significa que el responder no tiene información sobre el certificado requerido.

- ***CRS (Certificate Revocation System):*** CRS fue propuesto por Silvio Micali en 1995 [Mic05] .Aquí, la CA define n intervalos de actualización de duración i. Luego elige pseudo-aleatoriamente dos valores de 100 bits Y_0 y N_0, que son mantenidos en secreto por la CA, y una función de hash H apropiada. A continuación, calcula la cadena de hash $Y=Y_n=H^n(Y_0)$ y $N=H(N_0)$ y expide cada certificado con dos campos de extensión, de 100 bits cada uno, llamados Y ("*yes*") y N ("*no*"). La CA publica además periódicamente en el directorio una lista L con todos los certificados expedidos aún no expirados y los nuevos certificados expedidos en el intervalo i, cada uno con un valor V de 100 bits, donde $V=Y_{n-i}=H^{n-i}(Y_0)$ si el certificado no ha sido revocado y $V=N_0$ si el certificado fue revocado durante el intervalo i. Esta lista va firmada digitalmente por la CA y con sello de tiempo.

  El verificador comprueba la firma de la lista L. Luego determina si el número serial del certificado solicitado está en la lista. Si es así, calcula $H(V)$ y verifica si es igual a N. Si son iguales, el certificado ha sido revocado. De lo contrario, calcula $H^i(V)$ y examina si es igual a Y. Si coinciden el certificado es todavía válido.

- ***CRT (Certificate Revocation Tree):*** Los CRTs fueron propuestos por Paul Kocher en 1998 [Koc98]. Un CRT es un árbol de hash binario MHT (*Merkle Hash Tree*) [Mer89]donde cada hoja es el número de serie de un certificado revocado, los nodos son el valor hash de la concatenación de sus hijos y la raíz está firmada digitalmente por la CA.

  El verificador envía una solicitud con el número serial del certificado solicitado al directorio, y éste le devuelve: el valor hash de los nodos hermanos hasta la raíz y la raíz firmada. El verificador debe calcular el valor de hash de la raíz con la información que recibe del directorio y descifrar la raíz firmada. Si los valores obtenidos en las dos operaciones son iguales el certificado ha sido revocado.

- ***AD (Authenticated Dictionary)*:** Los ADs fueron propuestos por Naor y Nissim en 1998 [NN98]. Un AD es un árbol de hash de Merkle cuyas hojas son los certificados revocados organizados por número de serie, y los nodos se calculan aplicando una función hash a los valores de sus hijos. La raíz está firmada por la CA. El árbol es b2,3 lo que quiere decir que cada nodo interior puede tener 2 o 3 hijos, y todos los caminos de la raíz a la hoja tienen la misma longitud porque está balanceado.

  El verificador envía una solicitud con el número serial del certificado solicitado al directorio, y éste le devuelve: el valor hash de los nodos hermanos hasta la raíz y la raíz firmada. El verificador debe calcular el hash de la raíz con la información que recibe del directorio y descifrar la raíz firmada. Si los valores obtenidos son iguales, el certificado ha sido revocado.

Los recursos limitados de los dispositivos móviles complican el uso de los mecanismos de revocación estándar basados en repositorios y responders. Con CRL se requiere que el terminal disponga de una gran capacidad de almacenamiento para guardar las listas de certificados revocados, y el envío de esta gran cantidad de información no es conveniente debido al ancho de banda limitado. OCSP supone por su parte un mayor coste computacional debido a las operaciones de validación que deben hacerse cuando se consulta el estado de revocación de uno o más certificados. Por otra parte, algunos dispositivos no soportan dos

comunicaciones simultáneas, lo que significaría que se debe interrumpir la comunicación cada vez que se valide un certificado.

Por ello, la tecnología WAP no incluye el concepto de revocación, y ha optado por emitir certificados de corta duración (48 horas). La gestión de estos certificados puede traer problemas porque existe un período en el que el nuevo certificado de corta duración y el anterior son a la vez válidos. Por eso es necesario que los dispositivos inalámbricos mantengan sincronizado su reloj y que preferiblemente conozcan la zona horaria, ya que la validez de los certificados es expresada normalmente en tiempo UTC (*Universal Time Coordinated*). Por otra parte, una política basada en el uso de certificados de corta duración podría exponer al servidor a ataques de negación del servicio (p.e. si el atacante envía muchas solicitudes de certificación al servidor) [WAP01b].

2.5.2 Validación de Caminos de Delegación

Un *Camino de Delegación* es una lista de ACs, enlazados por los nombres de sus emisores y propietarios. El verificador de privilegios deberá validar dicho camino comprobando que cada AA tiene los privilegios y la autorización suficiente para la delegación. Así, la validación de un camino de delegación incluye:

- Establecer un camino de delegación confiable.
- Autenticar a las entidades que hacen parte del camino de delegación. Para ello se usa el servicio de autenticación de una PKI [ITU00], validando los caminos de certificación de cada entidad que hace parte del camino de delegación.
- Verificar la firma digital de cada AC en el camino de delegación.
- Comprobar que los ACs no están caducados o han sido revocados por sus emisores. En PMI se utilizan las Listas de Revocación de Certificados de Atributos (ACRLs – *Attribute Certificate Revocation Lists*), que poseen el mismo formato y administración que las CRLs.
- Verificar que cada emisor estaba autorizado a delegar los privilegios afirmados.

- Confirmar que los privilegios en el AC del asertor de privilegios son suficientes cuando se les compara con la política de control de acceso.

2.5.2.1 Algoritmos de Búsqueda de Caminos

Tuomas Aura [Aur97] describió y comparó varios algoritmos para tomar decisiones de autorización a partir de una base de datos de certificados. Estos algoritmos están basados en las técnicas de búsqueda de caminos en grafos.

Aquí, un certificado es una tupla <*issuer, subjects, k, authorization*>, donde: *issuer* es el emisor del certificado, *subjects* es un conjunto de *n* entidades a quienes se les ha expedido dicho certificado (sujetos del certificado) y *k* es el valor umbral que determina cuantos sujetos pueden usar, y delegar posteriormente, los derechos especificados en *authorization.* Si el certificado tiene más de un sujeto se llama certificado de delegación conjunta.

La red de delegación puede visualizarse como un grafo, donde los certificados son arcos y las entidades son nodos. Así, una red de delegación autoriza a un cliente *c* a realizar una operación en nombre de un servidor *s* si se puede construir un árbol finito donde:

1. Cada nodo del árbol corresponde a una entidad. El nodo raíz es *s* y los nodos hojas son clientes *c*. La misma entidad puede corresponder a múltiples nodos.
2. Todos los hijos de los nodos están marcados con un certificado. El emisor del certificado debe ser la entidad de ese nodo y los hijos deben ser los sujetos del certificado. El número de hijos debe ser menor o igual que el valor umbral del certificado. El mismo certificado puede ser usado por múltiples nodos.

Los algoritmos creados tienen dos características a tenerse en cuenta: primero, se basan en algoritmos de búsqueda de caminos en grafos directos, por lo que no se consideran datos calculados con anterioridad, aunque algún tipo de almacenamiento podría mejorar su eficiencia; segundo, la verificación de las firmas no es parte de los algoritmos. Los algoritmos son:

- ***Depth-first Search Forward*:** Traza el flujo de los derechos de acceso desde el servidor hasta el cliente. Cuenta los caminos válidos que conducen desde los sujetos del certificado al cliente. Si la cuenta alcanza el umbral requerido por el certificado, existe un camino de autorización válido desde el emisor del certificado al cliente. Desafortunadamente el número de caminos en el grafo crece exponencialmente con el tamaño del grafo.
- ***Depth-first Search Backward:*** Realiza la búsqueda desde el cliente hasta encontrar al servidor, lo cual resulta ser más rápido. Cuenta el número de caminos que conducen del cliente a los sujetos del certificado, y cuando el valor umbral es alcanzado continúa la búsqueda hacia atrás desde el emisor del certificado de delegación conjunta.
- ***Breadth-first Search Backward:*** Encuentra los servidores incrementando la distancia desde el cliente. Sin embargo, no se toma en cuenta el emisor del certificado de delegación conjunta hasta que se hayan encontrado suficientes sujetos del certificado.
- ***Two-way Search:*** Inicia la búsqueda tanto desde el servidor como desde el cliente marcando los nodos visitados a lo largo del camino. Cuando una búsqueda encuentra un nodo visitado por la otra, se establece que existe un camino. Para completar el camino buscado, la búsqueda que había visitado primero el nodo debe retroceder y encontrar el nodo otra vez, a menos que el sistema memorice los caminos de todos los nodos visitados.

Las pruebas mostraron que el algoritmo más eficiente es el de búsqueda bidireccional (two-way search). Éste marca una o dos entidades certificadas por el servidor con la búsqueda hacia adelante y luego trata de localizar uno de los nodos marcados con la búsqueda hacia atrás. Por tanto, es un poco más rápido que los algoritmos de búsqueda hacia atrás (search backward) pero más difícil de implementar. El algoritmo de búsqueda bidireccional es mejor cuando no existen certificados de delegación conjunta mientras que los de búsqueda hacia atrás los manejan particularmente bien.

2.5.3 Estándares para Validación de Certificados

La recomendación X.509 [ITU00] expresa el concepto de camino de certificación y describe los requisitos de este tipo de caminos.

El RFC 2459 [HPF99], más tarde reemplazado por el RFC 3280 [HPF02], contiene un algoritmo para validar caminos de certificación y define el conjunto de variables necesarias para llevar a cabo este proceso.

Por otra parte, el RFC 3379 [PH02] especifica los requerimientos para la validación de caminos de delegación (DPV – *Delegation Path Validation*) y el descubrimiento de este tipo de caminos (DPD – *Delegation Path Discovery*).

2.6 Conclusiones

En este capítulo se han analizado ampliamente los conceptos de autenticación y autorización, describiendo las diferentes técnicas utilizadas para brindar estos servicios de seguridad.

Además, se hace énfasis en las características de PKI y PMI como infraestructuras de autenticación y autorización, respectivamente, especificando las funciones de las entidades que las conforman y la forma en que interoperan entre sí.

También se describe el proceso de validación de caminos, detallando los pasos que lo componen y las soluciones y estándares propuestos para llevarlo a cabo. De aquí se evidencia la complejidad del proceso de validación tanto desde el punto de vista computacional, ya que los algoritmos de clave pública requieren cálculos matemáticos complejos, como considerando el conjunto de recursos necesarios para obtener los certificados, guardarlos y verificarlos. Así, un verificador con capacidades limitadas puede tener dificultades para llevar a cabo este proceso, especialmente si los caminos validados son largos. Por esa razón, es conveniente evaluar que tan críticos resultan ser parámetros como el coste computacional y la capacidad de almacenamiento cuando un terminal de poca capacidad, como una PDA o un teléfono móvil, realiza el proceso de validación de caminos.

Otro aspecto que dificulta la tarea del verificador, es la búsqueda o construcción de caminos en arquitecturas donde las relaciones de confianza son bidireccionales, ya que todas las opciones no llevan a la entidad objetivo y pueden existir múltiples caminos entre dos entidades. Por tanto, es conveniente proponer nuevas alternativas que contribuyan a simplificar la construcción de caminos en este tipo de modelos de confianza.

Capítulo 3

Coste Computacional de Validación de Caminos de Certificación y Delegación

3.1 Introducción 65
3.2 Definiciones y Suposiciones 66
3.3 Coste Computacional de Caminos de Certificación 67
3.4 Coste Computacional de Caminos de Delegación 73
3.5 Conclusiones 77

3.1 Introducción

El capítulo 2 describió el proceso de validación de caminos e identificó los aspectos de este proceso que dificultan la tarea del verificador y requieren ser mejorados. Uno de estos aspectos es el coste computacional, que puede ser elevado para verificadores con capacidad de procesamiento limitada, especialmente cuando los caminos son largos. En [SF04] y [SPF07] evaluamos el coste computacional del proceso de validación de caminos de certificación y en [SPF05a] el coste de los caminos de delegación, con el fin de determinar qué tan crítico puede llegar a ser este parámetro para los verificadores. El contenido de estos artículos se expresa a lo largo del presente capítulo.

La sección 3.2 da el concepto de coste computacional y las suposiciones en las que se basan los cálculos realizados. En la sección 3.3 calculamos el coste computacional de los caminos de certificación determinando el número de operaciones criptográficas que realiza el

verificador durante dicho proceso. Del mismo modo, en la sección 3.4 se evalúa el coste de los caminos de delegación. Finalmente, la sección 3.5 concluye.

3.2 Definiciones y Suposiciones

En la validación de caminos se realizan varios tipos de operaciones: operaciones criptográficas, operaciones de búsqueda en listas y operaciones de transmisión de información; pero quizás las primeras son las que demandan mayor capacidad de cálculo al verificador debido a su complejidad. Por este motivo, calculamos el coste computacional, del proceso de validación de caminos de certificación y de delegación, teniendo en cuenta sólo las operaciones criptográficas. Son operaciones criptográficas las de hash, cifrado, descifrado, firma y verificación. Sin embargo, las operaciones criptográficas que se llevan a cabo durante una validación de caminos son básicamente operaciones de hash y de verificación. Por tal motivo, definimos el coste computacional del verificador como el tiempo de CPU necesario para realizar las operaciones de hash y de verificación involucradas en la validación de un camino. La ecuación (3.1) especifica como se calcula este coste y la Tabla 3.1 contiene la notación utilizada en este capítulo.

$$COST = (OP_{hash} * T_{hash}) + (OP_{ver} * T_{ver}) \quad \textbf{(3.1)}$$

Aunque la mayoría de los estudios existentes sobre mecanismos de revocación de certificados se centran en la generación, distribución y actualización de la información de revocación desde el punto de vista del servidor [Coo99] [MF02], es importante determinar la incidencia de estos mecanismos en la capacidad de procesamiento del verificador, ya que son parte constitutiva del proceso de validación de caminos. Por esta razón, los cálculos de coste computacional realizados en este capítulo se hacen considerando 3 casos: sin revocación, con CRL como mecanismo de revocación y con OCSP como mecanismo de revocación.

Tabla 3.1: Notación de coste computacional

NOTACIÓN	SIGNIFICADO
COST	Coste computacional
OP_{hash}	Número de operaciones de hash
OP_{ver}	Número de operaciones de verificación
T_{hash}	Tiempo de ejecución de una operación de hash
T_{ver}	Tiempo de ejecución de una operación de verificación
L_d	Longitud de un camino de delegación
L_c	Longitud de un camino de certificación
R_d	Número de repositorios/responders consultados para verificar el estado de revocación de los certificados que hacen parte de un camino de delegación
R_c	Número de repositorios/responders consultados para verificar el estado de revocación de los certificados que hacen parte de un camino de certificación
$size_{CRL}$	Tamaño del contenido de una CRL [bytes]
$size_{OCSP}$	Tamaño del contenido de una respuesta OCSP [bytes]
N	Número de certificados emitidos por una autoridad

Ya que la criptografía de curvas elípticas brinda grandes ventajas con respecto a otros algoritmos de clave pública como DSA o RSA, especialmente en entornos donde los recursos son limitados [Van03], comparamos el coste computacional de la validación de caminos de certificación cuando se utiliza RSA de 1024 bits con el coste cuando se usa ECC de 163 bits, pues estos algoritmos brindan niveles de seguridad equivalentes [EC05]. Por otra parte, en el cálculo del coste de los caminos de delegación sólo consideramos RSA-1024. En todos los casos utilizamos SHA-1 como función de hash.

3.3 Coste Computacional de Caminos de Certificación

Para calcular el coste computacional de los caminos de certificación, requerimos el tiempo promedio de ejecución de las operaciones de verificación y de hash. Para SHA-1 y verificación con RSA-1024 tomamos los datos de una PDA Compaq iPAQH3630 con procesador StrongARM a 206MHz y sistema operativo Windows CE Pocket PC 2002 [AVT04]. Para la verificación con ECDSA de163 bits tomamos los datos de una PDA Linux con procesador StrongARM a 200MHz [GGC02]. Ya que estas PDAs poseen el mismo tipo

de procesador podemos tomar sus datos como base para nuestra comparación. Los tiempos se especifican en la Tabla 3.2.

Tabla 3.2: Tiempo de operaciones criptográficas en una PDA con procesador StrongARM a 200MHz

ALGORITMO	TIEMPO DE EJECUCIÓN
SHA –1	T_{hash} = 0,19 ms/operación
Verificación RSA	T_{ver} = 5,01 ms/operación
Verificación ECDSA	T_{ver} = 46,50 ms/operación

3.3.1 Primer Caso: Sin Revocación

En este caso, el verificador sólo realiza operaciones criptográficas cuando comprueba la firma de los certificados en el camino. Cada verificación de firma digital involucra una operación de hash y una de verificación. Por tanto, si el número de certificados en el camino es L_c, el verificador realiza L_c operaciones de hash y L_c operaciones de verificación en total (ver Tabla 3.3). Utilizando esta información, la ecuación (3.1) y los datos de la Tabla 3.2 calculamos el coste computacional del verificador especificado en la Tabla 3.4.

3.3.2 Segundo Caso: Con Revocación

Evaluamos ahora el coste computacional de la validación de un camino de certificación utilizando la misma PDA considerada en el caso anterior pero considerando esta vez los mecanismos de revocación estándar.

3.3.2.1 Revocación con CRL

En este caso, el verificador obtiene periódicamente una CRL que contiene la información de revocación de los certificados expedidos por su CA de confianza, y la almacena en su caché. El verificador debe comprobar la firma de dicha CRL antes de confiar en su contenido; por tanto, si todos los certificados en el camino de certificación que se está validando fueron expedidos por la misma CA de confianza, es decir, pertenecen al dominio de certificación del

verificador, su información de revocación estará contenida en la CRL obtenida periódicamente por éste. Por ende, si el proceso de validación se realiza antes de la siguiente actualización de la CRL, el verificador ya habrá comprobado la firma de la CRL en el momento en que inicia la validación y sólo realizará las operaciones criptográficas necesarias para verificar la firma de los certificados en el camino, es decir, L_c operaciones de hash y L_c de verificación, si L_c es la longitud del camino.

Sin embargo, cuando los certificados no pertenecen al mismo dominio de certificación del verificador, éste debe comprobar la firma de las CRLs que contienen la información de revocación de los certificados en el camino y además validar el camino de certificación de los emisores de dichas CRLs (por motivos de simplicidad, se omiten las operaciones criptográficas derivadas de la validación de estos caminos). Cada verificación de firma de una CRL implica una operación de hash y una de verificación. El número total de operaciones criptográficas dependerá del número de CRLs (R_c) que deba consultar el verificador (repositorios consultados) para obtener la información de revocación de todos los certificados en el camino, siendo L_c el número máximo posible de CRLs consultadas durante la validación de un camino (una por cada certificado en el camino), por tanto, $0 \leq R_c \leq L_c$. Así, el verificador realizará L_c+R_c operaciones de verificación: L_c operaciones para verificar la firma de los certificados y R_c operaciones para verificar la firma de las CRLs (ver Tabla 3.3). Utilizando esta información, la ecuación (3.1) y los datos en la Tabla 3.2 se obtiene el coste computacional especificado en la Tabla 3.4 para este caso.

3.3.2.2 Revocación con OCSP

Cuando un solo responder OCSP brinda la información de revocación de todos los certificados en el camino, además de las operaciones criptográficas necesarias para verificar la firma de cada certificado (L_c operaciones de hash y L_c de verificación), el verificador debe realizar dos operaciones de hash (sobre el DN y la clave pública del emisor) por certificado para construir la solicitud que envía al responder OCSP, es decir, en total $2L_c$ operaciones de hash. Además, debe verificar la firma de la respuesta que le retorna el responder OCSP, lo que implica una operación de hash y una de verificación adicional (por simplicidad se omiten las

operaciones criptográficas derivadas del chequeo de validez del certificado del responder OCSP).

El número de operaciones criptográficas crece cuando el verificador consulta un mayor número de responders OCSP para conocer el estado de revocación de todos los certificados en el camino, pues esto implica verificar la firma de varias respuestas OCSP. Así, el número total de operaciones criptográficas dependerá del número de responders R_c a los que se les envíe una solicitud OCSP, donde L_c es el número máximo de responders consultados (uno por cada certificado en el camino) y 1 es el mínimo ya que se debe enviar al menos una solicitud OCSP, por tanto, $1 \leq R_c \leq L_c$ en este caso. El verificador realizará finalmente R_c operaciones de hash y R_c operaciones de verificación adicionales para verificar la firma de las respuestas OCSP que reciba (ver Tabla 3.3). Utilizando esta información, la ecuación (3.1) y los datos de la Tabla 3.2, se obtiene el coste computacional especificado en la Tabla 3.4 para este caso.

Tabla 3.3: Número de operaciones criptográficas en caminos de certificación

MECANISMO DE REVOCACIÓN	TIPO DE OPERACIÓN	
	OP_{hash}	OP_{ver}
Ninguno	L_c	L_c
CRL	$L_c + R_c$	$L_c + R_c$
OCSP	$3L_c + R_c$	$L_c + R_c$

Tabla 3.4: Coste computacional de validación de caminos de certificación

MECANISMO DE REVOCACIÓN	COSTE COMPUTACIONAL	
	RSA	ECDSA
Ninguno	$5{,}20*10^{-3}*L_c$	$46{,}69*10^{-3}*L_c$
CRL	$5{,}20*10^{-3}*L_c + 5{,}20*10^{-3}*R_c$	$46{,}69*10^{-3}*L_c + 46{,}69*10^{-3}*R_c$
OCSP	$5{,}58*10^{-3}*L_c + 5{,}20*10^{-3}*R_c$	$47{,}07*10^{-3}*L_c + 46{,}69*10^{-3}*R_c$

En las Figuras 3.1 y 3.2 se muestra el coste computacional del verificador, sin y con revocación, a medida que aumenta la longitud del camino de certificación L_c, y considerando el valor mínimo y máximo de R_c para cada caso. El algoritmo de clave pública utilizado en la Figura 3.1 es RSA, mientras que en la Figura 3.2 se usa ECDSA. La Figura 3.3 compara el máximo coste computacional obtenido en cada caso (L_c=10) con RSA y ECDSA. Las tablas de datos de estas figuras se encuentran en el apéndice A, Tablas A.1, A.2 y A.3.

Aún cuando, el coste computacional calculado en todos los casos es razonable para las PDAs consideradas, el procesador de estos dispositivos es usualmente mucho más potente y disponen de más memoria comparados con las tarjetas inteligentes y los teléfonos móviles, cuyo tiempo de ejecución de una operación de verificación es mucho mayor, como se puede apreciar en la Tabla 3.5, tomada de [TG04]. Así, en el caso del Nokia 6610, y considerando sólo las operaciones de verificación, el coste computacional con revocación y R_c=L_c=10 sería de 49,76 segundos, mientras que el Siemens S55 tardaría 613,22 segundos en validar el mismo camino, es decir, más de 10 minutos.

Tabla 3.5: Tiempo de operaciones de verificación con RSA-1937 en teléfonos móviles

DISPOSITIVO	PRIMERA EJECUCIÓN (ms)	PROMEDIO (ms)
Nokia 6610	2825	2488
Nokia 6600	157	139
Ericsson P900	109	97
Siemens S55	30094	30661

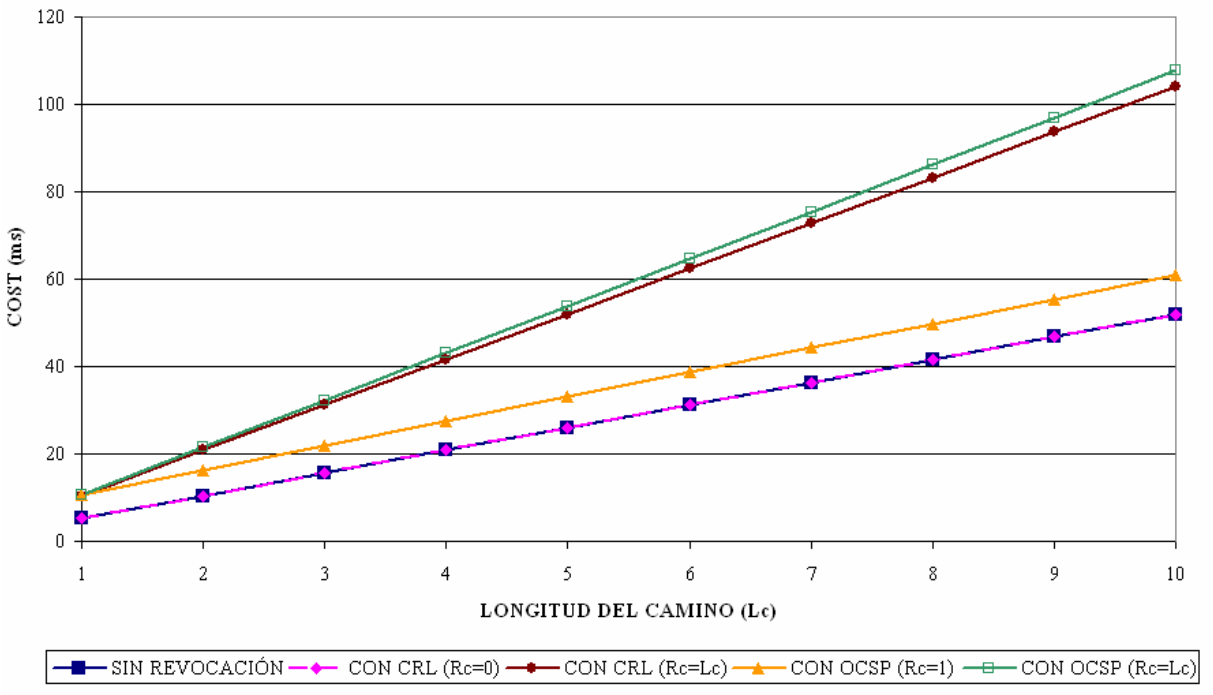

Figura 3.1: Coste computacional sin y con revocación usando RSA

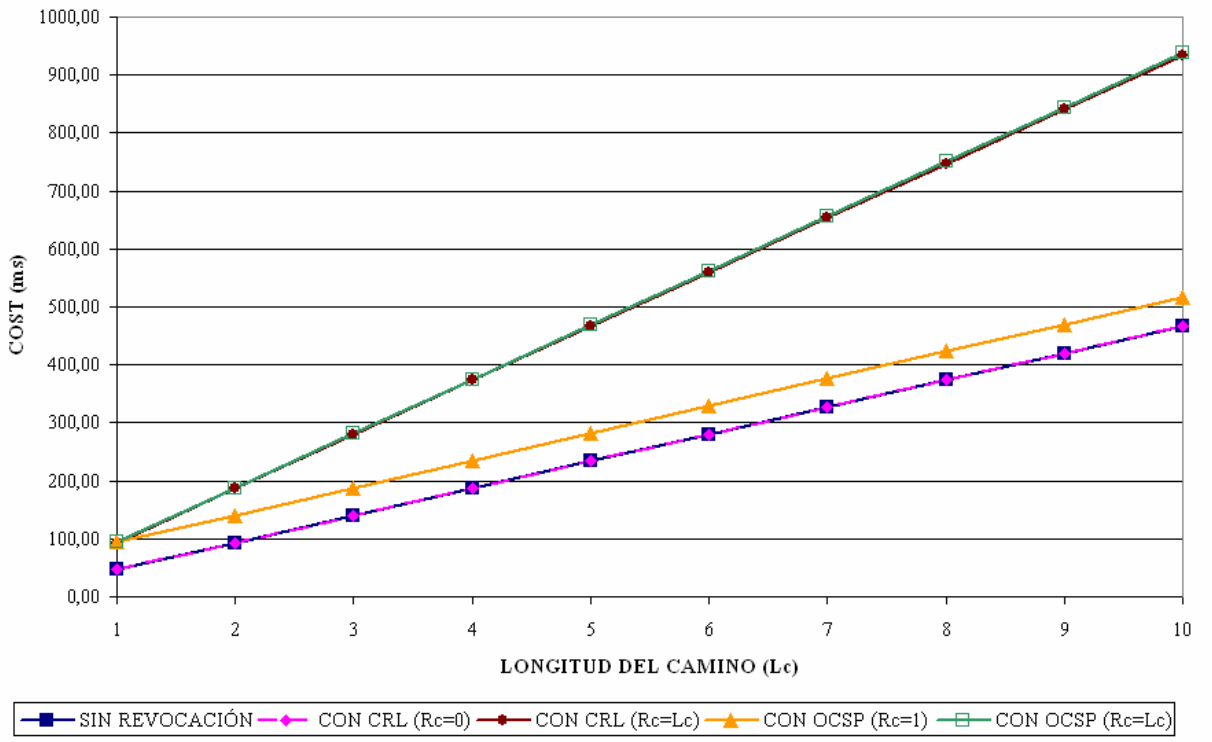

Figura 3.2: Coste computacional sin y con revocación usando ECDSA

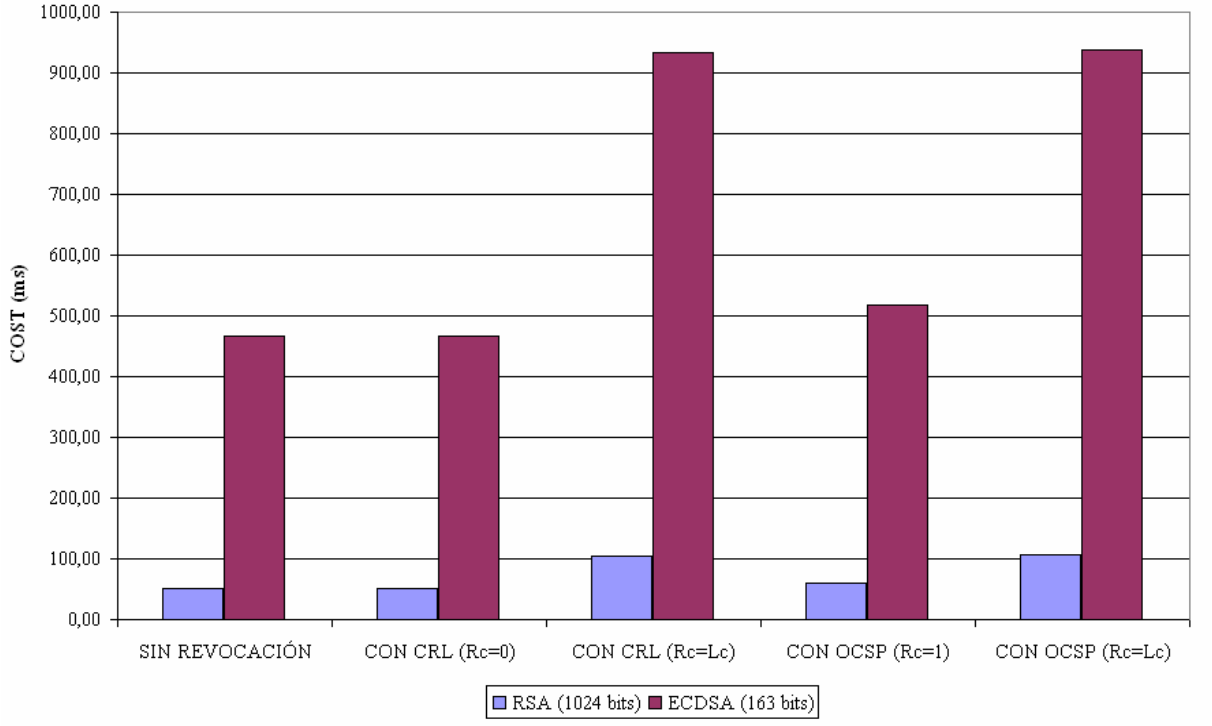

Figura 3.3: Coste computacional con RSA (1024 bits) y ECDSA (163 bits) para PDA con procesador StrongARM a 200MHz

3.4 Coste Computacional de Caminos de Delegación

En esta sección vamos a calcular el coste computacional de las operaciones criptográficas que realiza un verificador de privilegios durante la validación de un camino de delegación. Se denota como L_d la longitud del camino de delegación y suponemos que todos los caminos de certificación involucrados en este proceso de validación tienen la longitud L_c.

Ya que el verificador de privilegios suele ser un dispositivo de gran capacidad, como tiempos de ejecución de la función de hash SHA-1 y el algoritmo de clave pública RSA-1024 se tomaron los de un ordenador con procesador Pentium 4 a 2,1GHz bajo Windows XP SP1. Estos valores fueron codificados en C++ y compilados con Microsoft Visual C++ .NET 2003, según se especifica en [Dai04] (ver Tabla 3.6).

Tabla 3.6: Tiempo de operaciones criptográficas en ordenador con procesador Pentium 4 a 2,1GHz

ALGORITMO	TIEMPO DE EJECUCIÓN
SHA -1	$T_{hash=}$ 14,03 ns/byte
Verificación RSA-1024	$T_{ver=}$ 0,18 ms/operación

Como se aprecia en la Tabla 3.6, el tiempo de ejecución de una operación de hash va a depender del tamaño en bytes de la información a la que se aplica esta función. Por ello, nuestros cálculos se hacen tomando como referencia el tamaño de los campos del certificado y la CRL dados en los ejemplos C.2 y C.4 de [HPF02].

3.4.1 Primer Caso: Sin Revocación

En este caso, las operaciones criptográficas que realiza el verificador de privilegios son:

- Verificación de la firma digital de cada AC en el camino de delegación: L_d operaciones de hash, L_d operaciones de verificación.
- Verificación de la firma digital de los PKCs que hacen parte del camino de certificación de las entidades que conforman el camino de delegación: (L_d*L_c) operaciones de hash y (L_d*L_c) operaciones de verificación.

El número total de operaciones criptográficas que realiza el verificador en este caso se especifica en la Tabla 3.9.

Para calcular el tiempo de ejecución de las operaciones de hash, tomamos como tamaño del contenido de un PKC el especificado en el ejemplo C.2 de [HPF02], es decir, 665 bytes. Por otra parte, el tamaño del contenido de un AC, lo determinamos tomando como base el tamaño de los campos del PKC antes citado y la estructura de un certificado de atributos [ITU00] (omitiendo los campos opcionales *issuerUniqueID* y *extensions*, e incluyendo dos atributos de 27 bytes cada uno). La Tabla 3.7 muestra el tamaño de los campos considerados, por lo que finalmente el contenido del AC ocupará 181 bytes.

Con esta información, la ecuación (3.1) y los datos de la Tabla 3.6, calculamos el siguiente coste computacional (ver Tabla 3.10):

$$\text{COST} = ((L_d*181) + (L_d*L_c*665))*14{,}03*10^{-9} + (L_d + (L_d*L_c))*0{,}18*10^{-3} \qquad \textbf{(3.2)}$$

Tabla 3.7: Tamaño del contenido de un certificado de atributos

CAMPOS	TAMAÑO(Bytes)
version	3
holder(entityName)	42
issuer(issuerName)	42
signature	9
serialNumber	1
attCertValidityPeriod	30
attributes	54
TOTAL	**181**

3.4.2 Segundo Caso: Con Revocación

3.4.2.1 Revocación con CRL

Además de las operaciones criptográficas consideradas en el caso anterior, el verificador de privilegios comprueba la firma de las CRLs que contienen el estado de revocación de los certificados que hacen parte tanto del camino de delegación como de los caminos de certificación involucrados. En el mejor de los casos, el estado de revocación de todos los certificados que hacen parte de un camino estará en una CRL descargada previamente por el

verificador, por lo que no es necesario verificar su firma durante el proceso de validación. En el peor de los casos, se descargará una CRL por cada certificado en el camino, ya sea porque los certificados pertenecen a dominios de certificación diferentes o porque existen distintos puntos de distribución. Por tanto, el número de operaciones criptográficas dependerá del número de repositorios R_d y R_c que deba consultar el verificador para obtener la información de revocación de todos los certificados en los diferentes caminos, donde: $0 \leq R_d \leq L_d$ y $0 \leq R_c \leq L_c$. La Tabla 3.9 muestra cuantas operaciones de hash y de verificación (descifrado con clave pública) realiza el verificador de privilegios en este caso.

Para calcular el tiempo de ejecución de la operaciones de hash, se considera que las CRLs descargadas por el verificador de privilegios tienen el mismo tamaño de la CRL en el ejemplo C.4 de [HPF02]. Su contenido ocupa 140 bytes de los cuales 32 bytes corresponden a la información de revocación de un solo certificado. Si el 10% de los N certificados emitidos por una autoridad han sido revocados, el tamaño del contenido de la CRL es:

$$\text{size}_{\text{CRL}} = 108 + (32*0{,}1*\text{N}) = 108 + 3{,}2*\text{N} \qquad \textbf{(3.3)}$$

Utilizando esta información, la ecuación (3.1) y los datos de la Tabla 3.6, obtenemos el siguiente coste computacional (ver Tabla 3.10):

$$\begin{aligned}\text{COST} = &((L_d*181) + (L_d*L_c*665) + (R_d*\text{size}_{\text{CRL}}) + (L_d*R_c*\text{size}_{\text{CRL}}))*14{,}03*10^{-9} \\ &+(L_d + (L_d*L_c) + R_d + (L_d*R_c))*0{,}18*10^{-3}\end{aligned} \qquad \textbf{(3.4)}$$

3.4.2.2 Revocación con OCSP

Además de las operaciones criptográficas consideradas en el primer caso, para formar una solicitud OCSP, el verificador realiza dos operaciones de hash por certificado (sobre el DN y sobre la clave pública del emisor). Ya que la firma de estas solicitudes es opcional, suponemos que no van firmadas. En el mejor de los casos, el verificador hará una solicitud OCSP por todos los certificados en el camino, y en el peor de los casos, tendrá que hacer una solicitud OCSP por cada certificado, es decir, $1 \leq R_d \leq L_d$ y $1 \leq R_c \leq L_c$, donde R_d y R_c es el número de responders consultados para verificar el estado de revocación de los certificados. Además,

el verificador debe comprobar la firma de las respuestas OCSP que reciba. El número de operaciones criptográficas que realiza el verificador, en este caso, se especifica en la Tabla 3.9.

Teniendo como referencia el tamaño de los campos del certificado en el ejemplo C.2 de [HPF02], y conociendo los campos que conforman una respuesta OCSP básica [MAM99], se puede establecer que el contenido de dicha respuesta, omitiendo los campos opcionales, ocupa aproximadamente 141 bytes, de los cuales 74 bytes corresponden a la información de revocación de un certificado, como se observa en la Tabla 3.8.

Por tanto, el tamaño del contenido de una respuesta OCSP con la información de revocación de *L* certificados es:

$$size_{OCSP} = 67 + (74*L) \quad \textbf{(3.5)}$$

Adicionalmente, para formar una solicitud OCSP, se realiza una operación de hash sobre el campo *issuer* de cada certificado, cuyo tamaño es de 42 bytes [HPF02], y otra sobre la clave pública del emisor del certificado, que es de 128 bytes (1024 bits) en RSA-1024.

Tabla 3.8: Tamaño del contenido de una respuesta OCSP básica

CAMPOS	TAMAÑO(Bytes)
version	3
responderID (byName)	42
produceAt	13
hashAlgorithm	7
issuerNameHash	20
issuerKeyHash	20
serialNumber	1
certStatus (revocationTime)	13
thisUpdate	13
signatureAlgorithm	9
TOTAL	**141**

Con esta información, la ecuación (3.1) y los datos en la Tabla 3.6 calculamos el siguiente coste computacional (ver Tabla 3.10):

$$COST = ((42+128+181)*L_d + (42+128+665)*L_d*L_c + (size_{OCSP}*R_d) + (size_{OCSP}*L_d*R_c))*14{,}03*10^{-9} + (L_d + (L_d*L_c) + R_d + (L_d*R_c))*0{,}18*10^{-3} \quad \textbf{(3.6)}$$

Tabla 3.9: Número de operaciones criptográficas en caminos de delegación

MECANISMO DE REVOCACIÓN	TIPO DE OPERACIÓN	
	OP_{hash}	OP_{ver}
Ninguno	$L_d + (L_d*L_c)$	$L_d + (L_d*L_c)$
CRL	$L_d + (L_d*L_c) + R_d + (L_d*R_c)$	$L_d + (L_d*L_c) + R_d + (L_d*R_c)$
OCSP	$(3*L_d)+ (3*L_d*L_c) + R_d + (L_d*R_c)$	$L_d + (L_d*L_c) + R_d + (L_d*R_c)$

Tabla 3.10: Coste computacional de caminos de delegación

MECANISMO DE REVOCACIÓN	COSTE COMPUTACIONAL
Ninguno	$0{,}18*10^{-3}*L_d + 0{,}19*10^{-3}* L_d*L_c$
CRL	$0{,}18*10^{-3}*L_d + 0{,}19*10^{-3}*L_d*L_c +$ $(0{,}18*10^{-3} + 44{,}90*10^{-9}*N)*(R_d + (L_d*R_c))$
OCSP	$0{,}18*10^{-3}*L_d + 0{,}19*10^{-3}*L_d*L_c + 0{,}18*10^{-3}*R_d + 0{,}18*10^{-3}* L_d*R_c$

La Figura 3.4 muestra el coste computacional sin y con revocación a medida que aumenta la longitud del camino de delegación (L_d) y considerando los valores mínimo y máximo de R_c y R_d en cada caso. Se fija el parámetro L_c=3. La tabla de datos de esta figura se encuentra en el apéndice A, Tabla A.4.

3.5 Conclusiones

El proceso de validación de caminos le demanda al verificador gran cantidad de tiempo y recursos. Por ello, uno de los retos de PKI y PMI es la gestión eficiente de este tipo de caminos.

De las operaciones que se realizan durante dicho proceso, las criptográficas son las que quizás le exigen al verificador una mayor capacidad de cálculo. Por lo que en este capítulo determinamos la influencia de los mecanismos de revocación en el coste computacional del verificador teniendo en cuenta únicamente las operaciones criptográficas que realiza durante el proceso de validación de caminos tanto de certificación como de delegación.

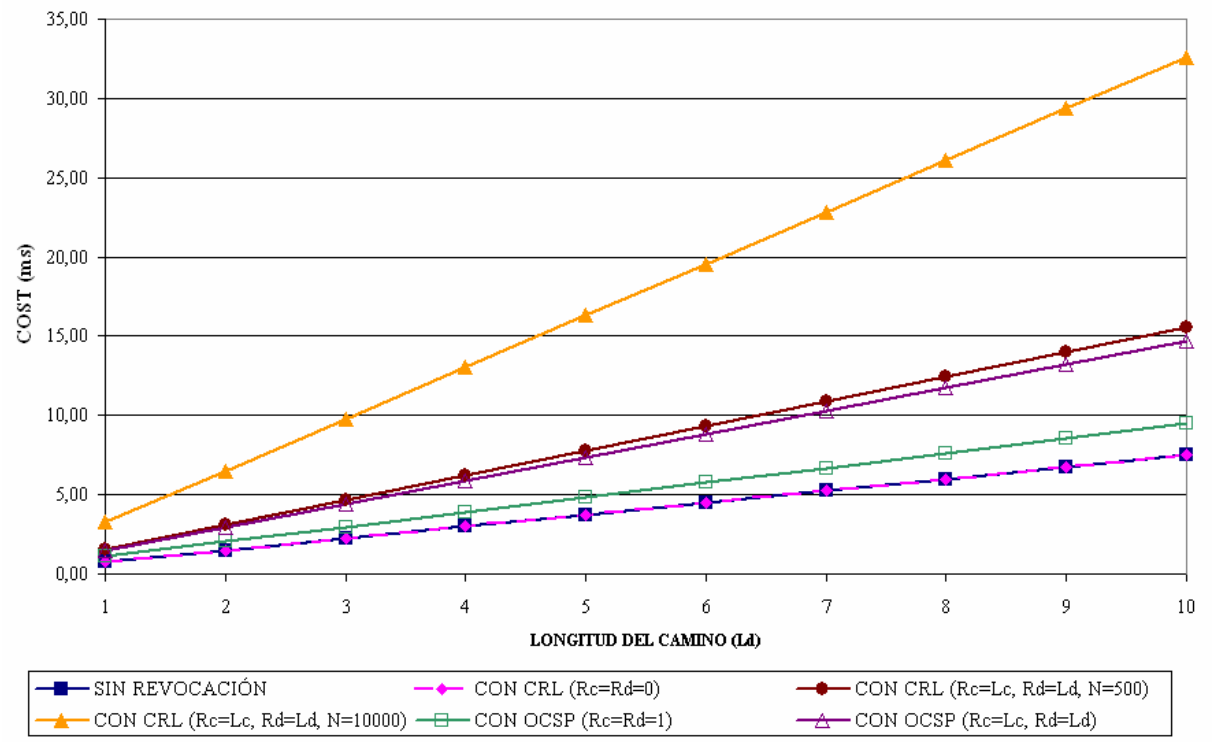

Figura 3.4: Coste computacional sin y con revocación de caminos de delegación

Las Tablas 3.3 y 3.9 muestran que cuando no se utiliza ningún mecanismo de revocación y con CRL, el número de operaciones de hash es igual al número de operaciones de verificación. Esto se debe a que sólo se realizan operaciones criptográficas cuando se verifica la firma de los certificados y las CRLs, y cada verificación de firma digital involucra una operación de hash y una de descifrado con clave pública. Con OCSP en cambio, el número de operaciones de hash es mayor que el número de operaciones de verificación, ya que una solicitud OCSP involucra dos operaciones de hash por cada certificado. Sin embargo, el mayor número de operaciones de hash no es un gran inconveniente para OCSP pues el tiempo de ejecución de estas operaciones es mucho menor que el de una operación de verificación (ver Tablas 3.2 y 3.6).

En el caso de la validación de caminos de certificación, utilizamos como verificador una PDA con procesador StrongARM a 200MHz. Según las Figuras 3.1 y 3.2, la diferencia entre

el coste de los caminos de certificación con OCSP (R_c=1) y el coste con CRL (R_c=0) se mantiene casi constante a medida que se incrementa la longitud del camino. Por otro lado, con R_c=L_c, y RSA la diferencia llega a ser notoria sólo para caminos de certificación muy largos (L_c>6), mientras con ECDSA se obtienen casi los mismos resultados con ambos mecanismos de revocación. Adicionalmente, en la Figura 3.3 se observa que el coste computacional con ECDSA es mucho mayor que el coste obtenido con RSA, lo que muestra que a pesar de los grandes beneficios que brinda ECDSA en entornos con recursos limitados, cuando se validan caminos RSA es una mejor alternativa.

En el caso de los caminos de delegación, utilizamos como verificador de privilegios un ordenador con procesador Pentium 4 a 2,1GHz. Aquí, el tiempo de ejecución de una operación de hash depende del tamaño de la información sobre la cual se aplique dicha operación (ver Tabla 3.6). El tamaño del contenido de las respuestas OCSP (ecuación (3.5)) depende sólo del número de certificados cuyo estado de revocación se esté consultando, por lo que es menor que el tamaño del contenido de una CRL (ecuación (3.3)), que depende del número de certificados revocados. Esto implica no sólo un mayor tiempo de ejecución de las operaciones de hash cuando se utiliza CRL sino también la necesidad de una mayor capacidad de almacenamiento.

En la Figura 3.4 se observa que el coste computacional de un camino de delegación con OCSP, es mayor que el coste con CRL cuando R_d y R_c toman su valor mínimo, debido a las operaciones criptográficas involucradas en la construcción de una solicitud OCSP y la verificación de la firma de su respectiva respuesta. Por otra parte, el coste computacional máximo con OCSP y CRL es muy similar cuando el número de certificados emitidos (N) es igual a 500, pero un incremento de N=10^4 hace que el coste computacional con CRL sea mayor que con OCSP. Lo mismo ocurre si se incrementa el porcentaje de certificados revocados, que en nuestros cálculos es siempre el 10% de N.

A pesar de que los tiempos encontrados para la PDA no consideran el tamaño de la información sobre la que se realizan las operaciones de hash, consideramos que este factor influye más en las PDAs que en un ordenador. Por lo que podemos concluir que, aunque el coste computacional de los caminos de certificación para la PDA con procesador StrongARM

a 200MHz y de los caminos de delegación para el ordenador con procesador Pentium 4 a 2,1GHz, es en todos los casos razonable, el mecanismo de revocación más apropiado, cuando el porcentaje de certificados revocados es alto y/o el número de certificados emitidos por las autoridades es grande, es OCSP.

Además, las PDAs tienen usualmente mucha más capacidad de procesamiento y memoria que los teléfonos móviles y las tarjetas inteligentes. Así que si se ejecutarán los algoritmos criptográficos en las tarjetas inteligentes, como lo sugieren las actuales especificaciones de seguridad (por ejemplo [WAP01f]), la capacidad de procesamiento sería un verdadero problema, ya que el procesador de las tarjetas inteligentes no puede manejar cálculos pesados. Adicionalmente, si comparamos la Tabla 3.2 con la Tabla 3.5, podemos ver que el tiempo de ejecución de una operación de verificación es mucho mayor para los teléfonos móviles que para la PDA considerada.

Por tanto, el coste computacional de la validación de caminos es un factor crítico cuando el verificador es un dispositivo con capacidades limitadas, al igual que otras características como la capacidad de almacenamiento, que será evaluada en el siguiente capítulo.

Capítulo 4

Capacidad de Almacenamiento en Validación de Caminos de Certificación

4.1 Introducción 81
4.2 Capacidad de Almacenamiento del Verificador 82
4.3 Ejemplo de Aplicación 90
4.4 Conclusiones 94

4.1 Introducción

En el capítulo 3 evaluamos el coste computacional de los caminos de certificación y de delegación, y se determinó que el tamaño de la información sobre la que se aplican las operaciones criptográficas influye en este coste. Por tal motivo, en el presente capítulo calculamos la capacidad de almacenamiento requerida por un verificador para realizar el proceso de validación de caminos de certificación. No consideramos esta vez los caminos de delegación, porque el verificador de privilegios suele ser un dispositivo de gran capacidad y lo que realmente nos interesa establecer es qué tan crítica es la capacidad de almacenamiento requerida para un verificador con poca capacidad. Los resultados expresados en este capítulo fueron publicados en los artículos [SPF05b] y [SPF05d].

En la sección 4.2 se calcula la capacidad de almacenamiento del verificador sin y con revocación. La sección 4.3 contiene un ejemplo en el que se determina la capacidad de

almacenamiento que debe tener un verificador en un caso específico. Finalmente, la sección 4.4 concluye.

4.2 Capacidad de Almacenamiento del Verificador

Para calcular la capacidad de almacenamiento con la que debe contar un verificador para realizar la validación de un camino de certificación, suponemos que cuando se utiliza RSA-1024, como algoritmo de clave pública, los certificados del verificador y la entidad objetivo son como el certificado de cliente del ejemplo D.1 en [WAP01g], que ocupa 425 bytes. Además se considera que los certificados de la base de confianza del verificador y demás CAs en el camino son como el certificado del ejemplo D.2 en [WAP01g], que ocupa 473 bytes. Estos certificados son de menor tamaño que el considerado en el capítulo anterior, pues para adaptar la tecnología PKI a los dispositivos móviles (WPKI) se limitó el tamaño de los PKCs. Por otra parte, cuando se utiliza ECDSA de 163 bits como algoritmo de clave pública, la clave pública y la firma de cada certificado ocuparán únicamente 21 bytes, a diferencia de RSA-1024 donde ocupan 128 bytes, por tanto, la diferencia de tamaño de los certificados entre un caso y otro será de 214 bytes, es decir, para ECDSA, los certificados del verificador y la entidad objetivo ocuparán 211 bytes, mientras los certificados de las CAs ocuparán 259 bytes. La Tabla 4.1 muestra la notación utilizada en este capítulo.

4.2.1 Primer Caso: Sin Revocación

Durante la validación de un camino de certificación, el momento en que el verificador tiene almacenada mayor cantidad de información es después de recuperar todos los certificados que hacen parte del camino. En ese momento, el verificador almacena la siguiente información en su memoria:

- Su propio certificado.

- El certificado de su base de confianza.
- Los certificados que hacen parte del camino: (L_c − 1) certificados correspondientes a las CAs intermedias y un certificado perteneciente a la entidad objetivo.

Tabla 4.1: Notación de capacidad de almacenamiento

NOTACIÓN	SIGNIFICADO
C	Capacidad de almacenamiento
L_c	Longitud de un camino de certificación
R_c	Número de repositorios/responders consultados para verificar el estado de revocación de los certificados que hacen parte de un camino de certificación
$size_V$	Tamaño del certificado del verificador [bytes]
$size_{RCA}$	Tamaño del certificado de la base de confianza del verificador [bytes]
$size_{CA}$	Tamaño del certificado de una CA intermedia [bytes]
$size_U$	Tamaño del certificado de la entidad objetivo [bytes]
Sig_{size}	Tamaño de la parte firmada de una CRL/respuesta OCSP [bytes]
CRL_{size}	Tamaño de una CRL [bytes]
RCert	Número de certificados revocados
N	Número de certificados emitidos por una autoridad
p	Porcentaje de certificados revocados
RequestOCSP	Tamaño de una solicitud OCSP [bytes]
ResponseOCSP	Tamaño de una respuesta OCSP [bytes]
$OCSP_{size}$	Espacio ocupado por todas las solicitudes y respuestas OCSP que envía y recibe un verificador durante el proceso de validación de caminos de certificación
PK_X	Clave pública de la entidad X
$CERT_X$	Certificado de la entidad X

Por simplicidad omitimos la información relacionada con las políticas de certificación y el conjunto de variables que se utilizan para realizar la verificación de firma y validez de los certificados, que ocupan poco espacio.

La capacidad de almacenamiento C que requiere el verificador en este caso, para poder realizar correctamente el proceso de validación de caminos, será la suma de la información mencionada, como se muestra en la ecuación (4.1). La Tabla 4.3 contiene la capacidad de almacenamiento requerida por el verificador cuando no se usa ningún mecanismo de revocación con RSA de 1024 bits y ECDSA de 163 bits.

$$C = \text{size}_V + \text{size}_{RCA} + \text{size}_{CA}*(L_c\text{-}1) + \text{size}_U \quad \textbf{(4.1)}$$

4.2.2 Segundo Caso: Con Revocación

Ahora evaluamos la capacidad de almacenamiento del verificador utilizando los mecanismos de revocación estándar.

4.2.2.1 Revocación con CRL

Cuando el mecanismo de revocación utilizado es CRL, además de la información considerada en el primer caso, el verificador debe recuperar y almacenar las CRLs que contienen el estado de revocación de los certificados en el camino. Para ello, tomamos como referencia la CRL del ejemplo C.4 en el RFC3280 [HPF02], cuyo contenido ocupa 140 bytes, de los cuales 32 bytes corresponden a la información de revocación de un certificado. Si *RCert* es el número de certificados revocados y Sig_{size} es el número de bytes que ocupa la parte firmada de la CRL, el tamaño de cada CRL (contenido + firma) es:

$$\text{CRL}_{\text{size}} = (32*RCert) + 108 + \text{Sig}_{\text{size}} \quad \textbf{(4.2)}$$

RCert será igual al número de certificados emitidos N por el porcentaje de certificados revocados *p*, tal como se expresó en la ecuación (3.3) del capítulo anterior.

$$RCert = \text{N}*p \quad \textbf{(4.3)}$$

Si todos los certificados en el camino pertenecen al mismo dominio de certificación del verificador, su información de revocación estará contenida en la CRL que el verificador recupera y verifica periódicamente, es decir, durante el proceso de validación sólo almacenaría una CRL. Sin embargo, cuando los certificados no pertenecen al mismo dominio de certificación del verificador, se deben recuperar y almacenar varias CRLs durante la validación de certificados. Por ello, suponemos que el número total de CRLs recuperadas por

el verificador durante el proceso de validación es R_c, donde L_c es el número máximo de CRLs a consultar, es decir, una por cada certificado en el camino ($1 \leq R_c \leq L_c$).

La capacidad de almacenamiento C que requiere el verificador en este caso se calcula utilizando las ecuaciones (4.1) y (4.2) como se especifica en la ecuación (4.4). En la Tabla 4.3 se expresa la capacidad de almacenamiento cuando se utiliza RSA y ECDSA, reemplazando Sig_{size} por el tamaño de la clave pública en cada caso.

$$C = size_V + size_{RCA} + size_{CA}*(L_c\text{-}1) + size_U + CRL_{size}*R_c \quad \textbf{(4.4)}$$

4.2.2.2 Revocación con OCSP

Cuando el mecanismo de revocación utilizado es OCSP, el verificador debe solicitar al responder la información de revocación de los certificados que hacen parte del camino y luego almacenar la respuesta OCSP que le informa sobre el estado de revocación de los mismos.

Podemos determinar el tamaño aproximado de una solicitud OCSP basándonos en su sintaxis [MAM99], y en el tamaño de los campos del certificado en el ejemplo C2 del RFC3280 [HPF02]. La Tabla 4.2 muestra los campos obligatorios de una solicitud OCSP conteniendo la información de un solo certificado, y especifica el tamaño aproximado de cada campo. Por otra parte, el tamaño aproximado del contenido de una respuesta OCSP ya fue especificado en la Tabla 3.8 (141 bytes), así que sólo restaría adicionarle el tamaño de la firma, que para RSA-1024 es de 128 bytes y para ECDSA de 163 bits es de 21 bytes.

Tabla 4.2: Tamaño de una solicitud OCSP

CAMPOS	TAMAÑO (Bytes)
version	3
hashAlgorithm	7
issuerNameHash	20
issuerKeyHash	20
serialNumber	1
TOTAL	**51**

De los 51 bytes que conforman una solicitud OCSP, 48 bytes corresponden a la información de un certificado. Por tanto, si se pregunta a un solo responder OCSP por el estado de revocación de los L_c certificados en el camino, el tamaño de la solicitud OCSP que se le envía será:

$$\text{RequestOCSP} = 3 + 48*L_c \quad \textbf{(4.5)}$$

Por otra parte, de los 141 bytes que conforman el contenido de una respuesta OCSP, 74 bytes corresponden al estado de revocación de un certificado. Por tanto, el tamaño de una respuesta OCSP conteniendo la información de revocación de los L_c certificados en el camino será:

$$\text{ResponseOCSP} = 67 + 74*L_c + \text{Sig}_{\text{size}} \quad \textbf{(4.6)}$$

Sin embargo, cuando los certificados no pertenecen al mismo dominio de certificación del verificador, éste debe solicitar a varios responder OCSP la información de revocación de los certificados en el camino, lo que ocasiona un mayor número de solicitudes y respuestas OCSP, de menor tamaño que las especificadas en las ecuaciones (4.5) y (4.6). Si el número de responders OCSP a los que se envía una solicitud es R_c ($1 \leq R_c \leq L_c$), el espacio ocupado por todas estas solicitudes y sus respectivas respuestas es:

$$\text{OCSP}_{\text{size}} = (3*R_c + 48*L_c) + (67*R_c + 74*L_c + \text{Sig}_{\text{size}}*R_c) = (70 + \text{Sig}_{\text{size}})*R_c + 122*L_c \quad \textbf{(4.7)}$$

La capacidad de almacenamiento que debe tener el verificador en este caso es por tanto la suma de las ecuaciones (4.1) y (4.7) como se especifica en la ecuación (4.8). La Tabla 4.3 muestra la capacidad de almacenamiento cuando se usa RSA y ECDSA.

$$C = \text{size}_{\text{V}} + \text{size}_{\text{RCA}} + \text{size}_{\text{CA}}*(L_c\text{-}1) + \text{size}_{\text{U}} + \text{OCSP}_{\text{size}} \quad \textbf{(4.8)}$$

Tabla 4.3: Capacidad de almacenamiento en validación de caminos de certificación

MECANISMO REVOCACIÓN	CAPACIDAD DE ALMACENAMIENTO	
	RSA	ECDSA
Ninguno	$850 + 473*L_c$	$422 + 259*L_c$
CRL	$850 + 473*L_c + 236*R_c + 32*RCert*R_c$	$422 + 259*L_c + 129*R_c + 32*RCert*R_c$
OCSP	$850+595*L_c + 198*R_c$	$422+381*L_c + 91*R_c$

La Figura 4.1 enseña la capacidad de almacenamiento requerida por el verificador en los diferentes casos cuando se utiliza RSA-1024, considerando los valores mínimo y máximo de R_c y $RCert$=100. Los resultados obtenidos para ECDSA de 163 bits se pueden observar en la Figura 4.2.

También evaluamos la capacidad de almacenamiento con CRL utilizando dos valores diferentes de *RCert* (100 y 1000). En la Figura 4.3 tenemos los resultados obtenidos para RSA y en la Figura 4.4 los resultados para ECDSA.

Finalmente, la Figura 4.5 compara la capacidad de almacenamiento obtenida en cada caso con L_c=10 para RSA y ECDSA.

Las tablas de datos de estas figuras se encuentran en el apéndice A, Tablas A.5, A.6, A.7, A.8 y A.9.

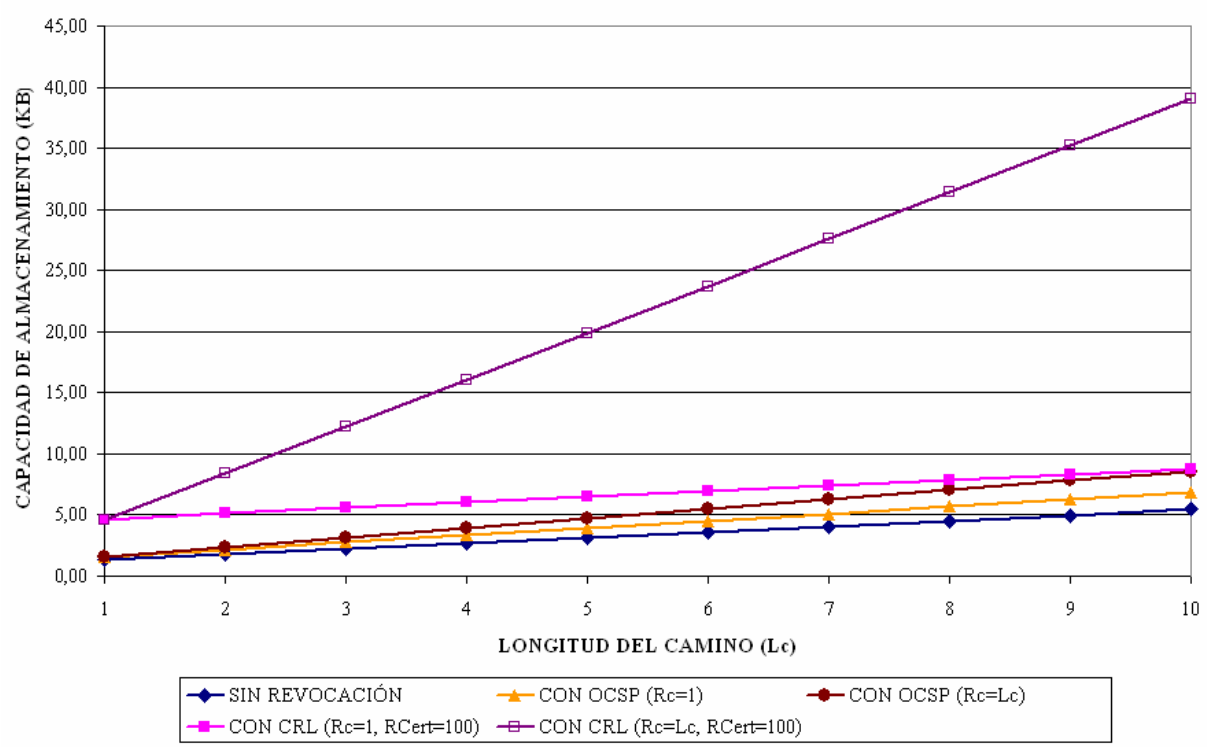

Figura 4.1: Capacidad de almacenamiento sin y con revocación usando RSA (1024 bits)

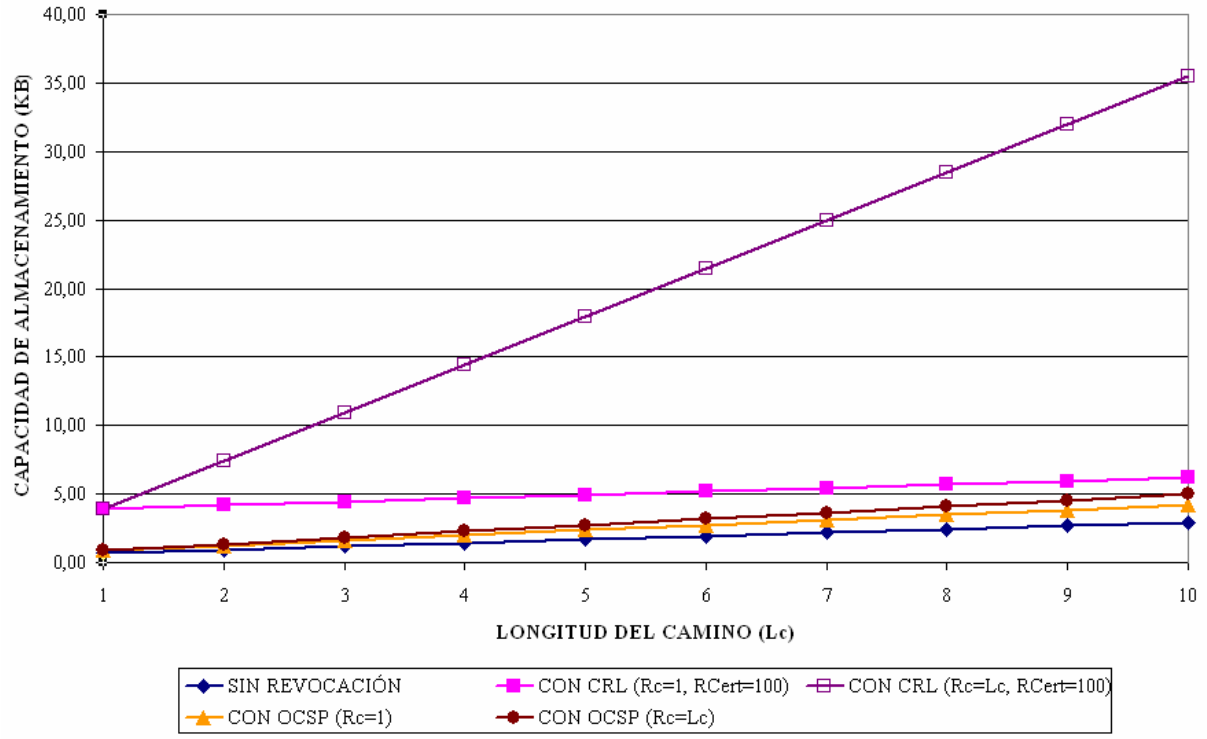

Figura 4.2: Capacidad de almacenamiento sin y con revocación usando ECDSA (163 bits)

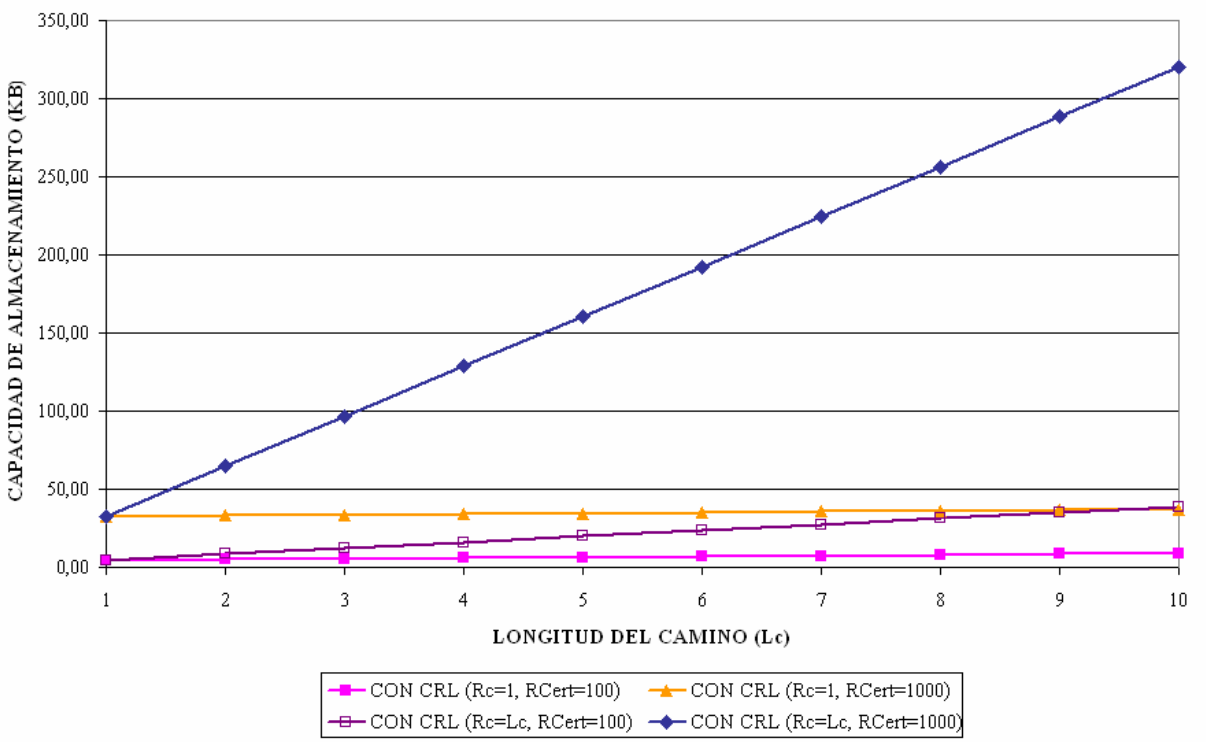

Figura 4.3: Capacidad de almacenamiento con CRL usando RSA (1024 bits)

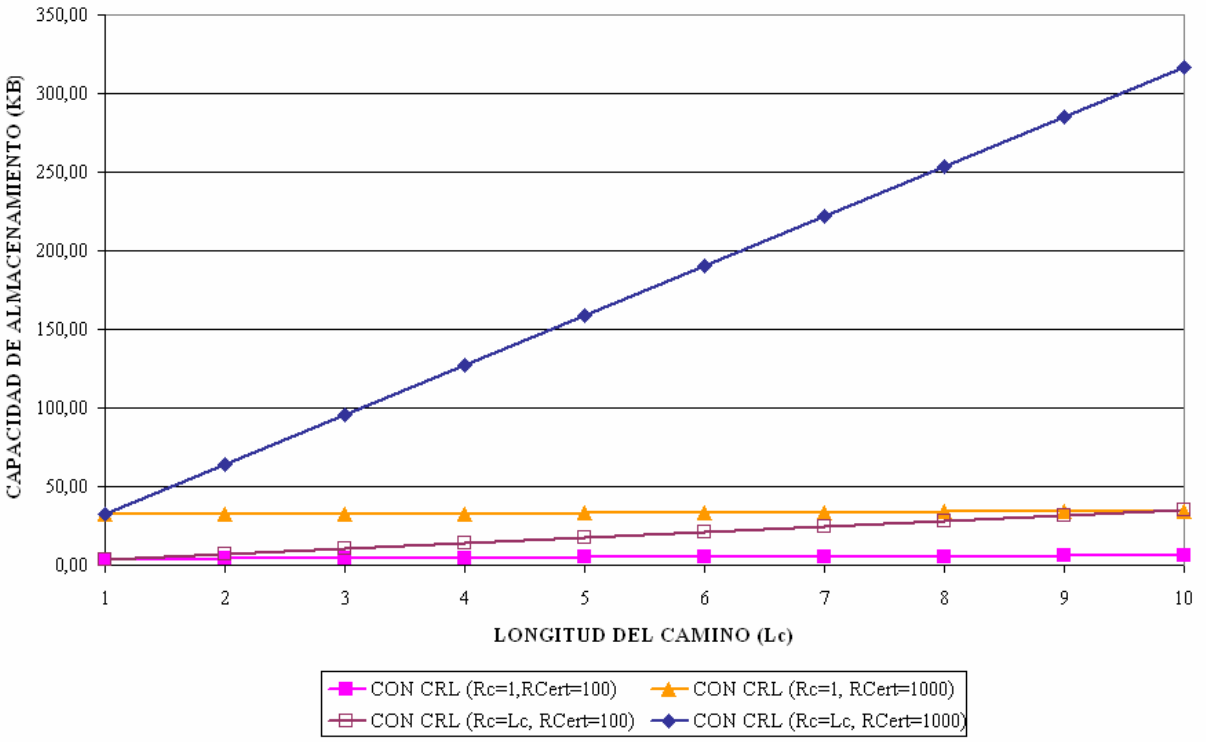

Figura 4.4: Capacidad de almacenamiento con CRL usando ECDSA (163 bits)

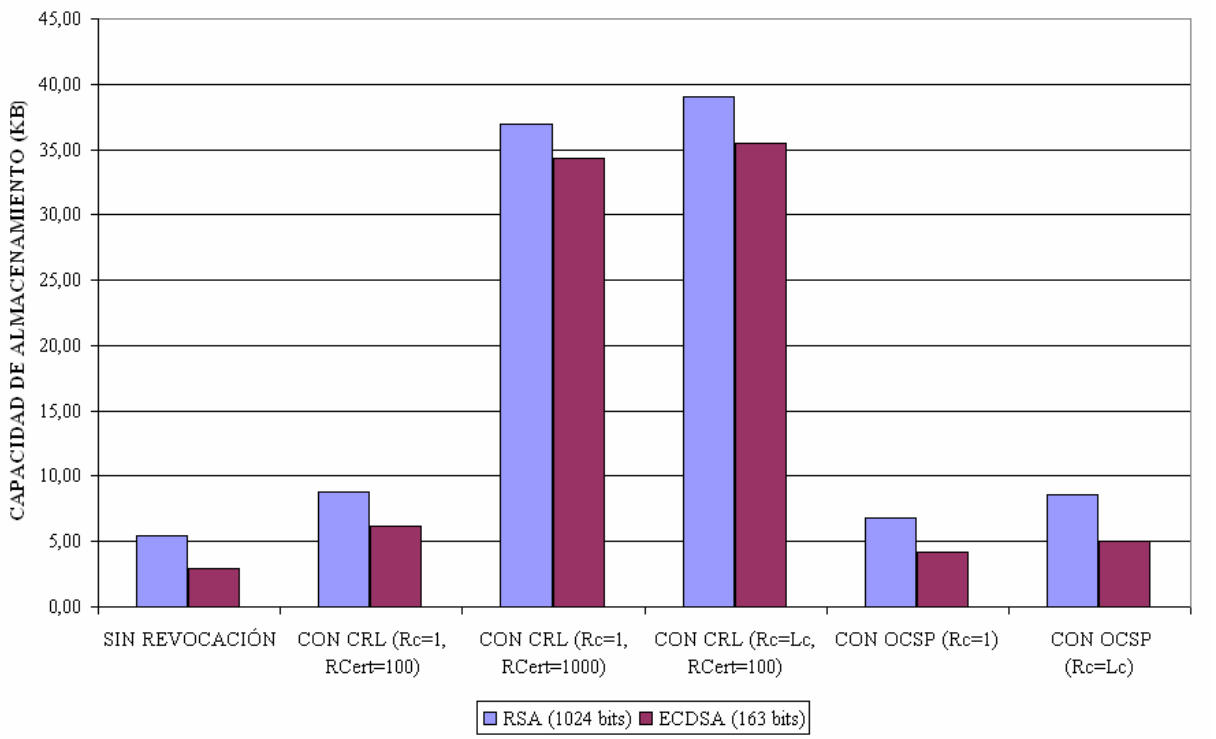

Figura 4.5: Capacidad de almacenamiento con RSA (1024 bits) y ECDSA (163 bits)

4.3 Ejemplo de Aplicación

La Figura 4.6 muestra parte de la arquitectura jerárquica empleada por Verisign [Ver05] conformada por:

- VR (*Verisign Root*) que certifica las claves públicas auto-firmadas de las autoridades de certificación primaria públicas (PCAs) y que es la base de confianza de la arquitectura, es decir, su clave pública PK_{VR} es conocida por todas las entidades que hacen parte de la jerarquía. El tamaño de esta clave pública es de 2048 bits cuando se usa RSA y 224 bits cuando se utiliza ECDSA [Van03].
- Dos PCAs (*Public Primary Certification Authorities*), PCA2 para los certificados clase 2 y PCA3 para los certificados clase 3. Estas PCAs son operadas por Verisign y sus claves públicas PK_{PCA2} y PK_{PCA3} son de 1024 bits para RSA y 163 bits en ECDSA.

- Dos CAs, una bajo cada PCA, cuyas claves públicas PK_{CA1} y PK_{CA2} son de 1024 bits para RSA y 163 bits para ECDSA. CA1 es una autoridad de certificación perteneciente a Verisign, mientras CA2 es una autoridad de certificación afiliada que actúa como centro de procesamiento [Ver03], de allí que posea su propio repositorio DIR2.
- Dos usuarios U3 y U2, U3 con certificado clase 3 y U2 con certificado clase 2 [Ver03].
- DIR es el repositorio de Verisign, donde se publican los certificados y la información de revocación de las autoridades que son propiedad de Verisign.

Las flechas representan los certificados entre las diferentes entidades que hacen parte de la arquitectura, mientras el círculo con flecha al lado de VR, PCA2 y PCA3, significa que estas autoridades se expiden sus propios certificados (certificados auto-firmados).

U3 es un usuario móvil que quiere enviarle un mensaje cifrado a U2, pero para ello necesita PK_{U2}. Vamos a calcular la capacidad de almacenamiento que debe tener el usuario U3 (verificador) para poder realizar el proceso de validación que le permitirá obtener la clave pública de U2, con y sin revocación.

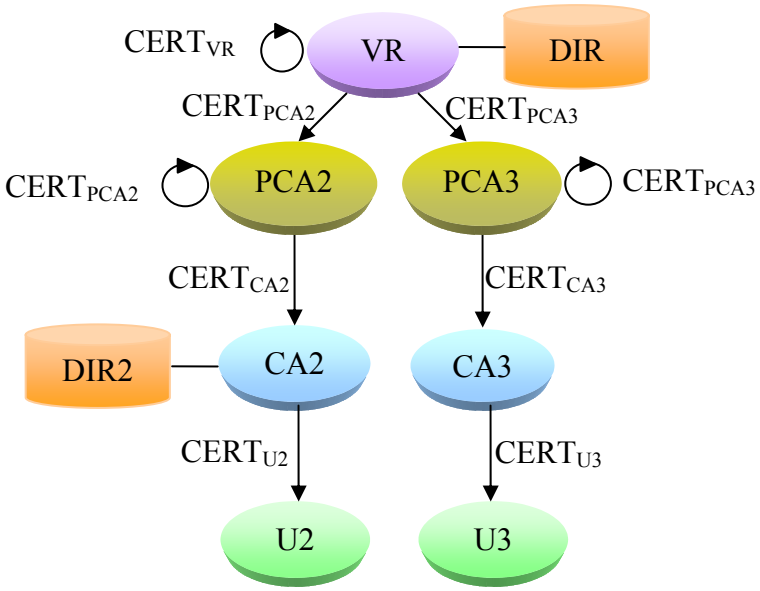

Figura 4.6: Arquitectura jerárquica de Verisign

El proceso de validación partirá de VR y el camino incluirá los certificados de PCA2, CA2 y U2, lo que quiere decir, que se deben recuperar y verificar tres certificados, es decir, L_c=3.

Suponemos que los certificados de todas las entidades en la jerarquía tienen el tamaño especificado en la sección 4.2.

4.3.1 Primer Caso: Sin Revocación

En este caso, U3 almacenará, además de su propio certificado $CERT_{U3}$ y el de su base de confianza $CERT_{VR}$, los certificados que hacen parte del camino, es decir, $CERT_{PCA2}$, $CERT_{CA2}$, $CERT_{U2}$. Es conveniente anotar, que la firma de $CERT_{PCA2}$ tiene un tamaño mayor puesto que se utiliza la clave pública de VR que es más grande. Adicionalmente, el tamaño de la clave pública y la firma de $CERT_{VR}$ también son mayores, ya que se trata de un certificado auto-firmado. La capacidad de almacenamiento del verificador se especifica en la Tabla 4.4 para RSA y ECDSA.

4.3.2 Segundo Caso: Con Revocación

En nuestro ejemplo, la información de revocación de los certificados $CERT_{PCA2}$ y $CERT_{CA2}$ se encuentra en el repositorio DIR y la del certificado $CERT_{U2}$ se encuentra en DIR2.

4.3.2.1 Revocación con CRL

Para verificar el estado de revocación de $CERT_{PCA2}$ y $CERT_{CA2}$, U3 utiliza la última CRL que descargó de DIR y para verificar el estado del certificado $CERT_{U2}$, tendrá que recuperar la última CRL publicada en DIR2, por tanto, R_c=2. La capacidad de almacenamiento del verificador en este caso suponiendo *RCert*=500, se especifica en la Tabla 4.4.

4.3.2.2 Revocación con OCSP

Aquí, U3 envía una solicitud OCSP con la información de los certificados $CERT_{PCA2}$ y $CERT_{CA2}$ a DIR y otra solicitud con la información del certificado $CERT_{U2}$ a DIR2, por tanto, R_c=2. La capacidad de almacenamiento del verificador con OCSP se especifica en la Tabla 4.4.

Tabla 4.4: Capacidad de almacenamiento en ejemplo de aplicación

MECANISMO REVOCACIÓN	CAPACIDAD DE ALMACENAMIENTO (Bytes)	
	RSA	ECDSA
Ninguno	2653	1283
CRL	35253	33548
OCSP	3543	1838

En la Figura 4.7 se muestra la capacidad de almacenamiento que requeriría el usuario U3 en los diferentes casos considerados por nuestro ejemplo de aplicación. La tabla de datos de esta figura se encuentra en el apéndice A, Tabla A.10.

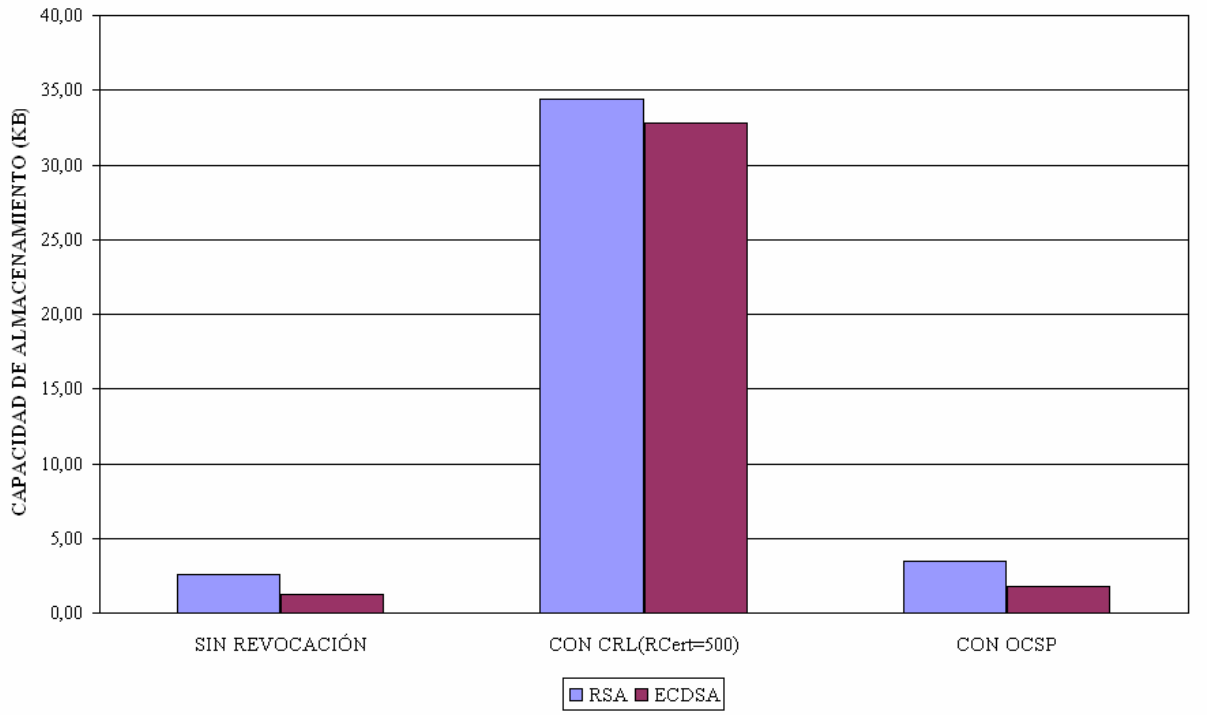

Figura 4.7: Capacidad de almacenamiento en ejemplo de aplicación

4.4 Conclusiones

Los dispositivos móviles poseen una capacidad de almacenamiento limitada, lo que no les permite realizar funciones complejas que requieran almacenar gran cantidad de datos. De allí que hallamos calculado en este capítulo la capacidad de almacenamiento que requiere un verificador durante el proceso de validación de caminos de certificación.

Las Figuras 4.1 y 4.2 muestran que la capacidad de almacenamiento requerida por el verificador cuando se utiliza OCSP es levemente mayor que la capacidad de almacenamiento cuando no se utiliza ningún mecanismo de revocación. Por otra parte, el verificador requiere una mayor capacidad de almacenamiento cuando utiliza CRL, la cual se ve influenciada en gran medida por el número de certificados revocados (*RCert*), como se observa en las Figuras 4.3. y 4.4.

Por otra parte, el uso de ECDSA como algoritmo de clave pública reduce la capacidad de almacenamiento requerida por el verificador, gracias a que se utilizan claves públicas de menor tamaño para ofrecer un nivel de seguridad comparable al ofrecido por RSA. Las Figuras 4.5 y 4.7 muestran que cuando no se utiliza ningún mecanismo de revocación, la reducción en la capacidad de almacenamiento ofrecida por ECDSA es del orden del 50%. En el caso de OCSP esta reducción es del orden del 40%, mientras que con CRL va disminuyendo a medida que crece el valor de *RCert*.

Adicionalmente, la capacidad de almacenamiento del verificador con CRL y *RCert*=1000 supera los 32 KB, que es la capacidad de muchas de las tarjetas SIM que poseen actualmente los teléfonos móviles. Por tanto, el proceso de validación de caminos de certificación por parte de este tipo de dispositivos no es siempre viable cuando se utiliza CRL. Así, OCSP no es sólo el mecanismo de revocación más apropiado por la capacidad de procesamiento que requiere, como se concluyó en el capítulo 3, sino también por la baja capacidad de almacenamiento demandada.

Nuestros esfuerzos se centraran de aquí en adelante en proponer nuevas alternativas que contribuyan a simplificar la labor del verificador en el proceso de validación de caminos.

Capítulo 5

TRUTHC (Trust Relationship Using Two Hash Chains)

5.1 Introducción 95
5.2 Validación desde el Punto de Vista del Verificador 96
5.3 Cadenas de Hash 98
5.4 Escenario de Estudio 99
5.5 Descripción de TRUTHC 103
5.6 Integración de TRUTHC con los Certificados X.509 108
5.7 Análisis de Seguridad 109
5.8 Evaluación 110
5.9 TRUTHC en Validación de Arquitecturas de Autorización 115
5.10 Conclusiones 116

5.1 Introducción

Los capítulos 3 y 4 mostraron que tanto el coste computacional como la capacidad de almacenamiento son factores críticos para un verificador con capacidades limitadas cuando se validan caminos de certificación. Es por ello conveniente buscar diferentes alternativas que simplifiquen el proceso de validación desde el punto de vista del verificador para acercar la tecnología PKI a los dispositivos limitados. Para reducir el coste computacional del verificador, en [SPF05c] y [SPF05e] proponemos TRUTHC (*Trust Relationship Using Two Hash Chains*), una forma alterna de establecer la relación de confianza entre las diferentes entidades de una PKI jerárquica utilizando cadenas de hash. De esta manera, disminuimos el

número de operaciones de descifrado con clave pública que debe realizar el verificador durante la validación de caminos. El contenido de estos artículos se expone a lo largo del presente capítulo.

En la sección 5.2 describimos las propuestas existentes que contribuyen a simplificar el proceso de validación de caminos desde el punto de vista del verificador. La sección 5.3 define lo que es una cadena de hash. El escenario en el que aplicamos nuestra propuesta se describe en la sección 5.4. La sección 5.5 explica la forma de establecer una relación de confianza alternativa entre las entidades de una PKI jerárquica utilizando TRUTHC. En la sección 5.6 nos centramos en la compatibilidad de TRUTHC con los certificados X.509. La sección 5.7 contiene un análisis de seguridad de nuestra propuesta. Luego, en la sección 5.8 se evalúa el coste computacional obtenido con nuestra propuesta y se compara con el de una PKI típica. La sección 5.9 describe como TRUTHC puede integrarse en las arquitecturas de autorización. Finalmente, la sección 5.10 concluye.

5.2 Validación desde el Punto de Vista del Verificador

En los últimos años, diversos autores han presentado varias propuestas que contribuyen a simplificar la labor del verificador en el proceso de validación de caminos de certificación.

Brian Hunter [Hun02] traslada las operaciones complejas de los clientes a los servidores. Para ello, propone que cada servidor cuente con una caché local que almacene los caminos de certificación previos y que dicha caché sea común a los clientes cuyos caminos tienen certificados y bases de confianza similares. De esta manera, aunque los caminos previos sólo coincidan parcialmente con el camino requerido, se disminuye el número de solicitudes a los repositorios y se evitan caminos falsos, reduciendo también el número de veces en que es construido el mismo camino. Sin embargo, el camino buscado no siempre está almacenado en la caché, por lo que en esos casos el servidor debe realizar todo el proceso de validación de caminos, que es bastante complejo.

Levi y Caglayan [LC99] proponen el uso de Certificados Anidados (*Nested Certificates*) para mejorar el desempeño y flexibilidad de los certificados clásicos, reduciendo el número de operaciones criptográficas necesarias para llevar a cabo el proceso de validación de caminos de certificación.

Un *Certificado Anidado* es un certificado de otro certificado, llamado *Certificado Sujeto*. Los certificados anidados contienen la firma y el hash realizados sobre el contenido del certificado sujeto, y van firmados digitalmente por una NCA (*Nested Certificate Authority*).

Para verificar un solo certificado sujeto, *sc*, se debe primero verificar la firma de su certificado anidado, *nc*, utilizando la clave pública de la NCA, y luego: 1) Recalcular el hash sobre el contenido del *sc* y compararlo con el guardado en el *nc*. 2) Comparar la firma del *sc* con la guardada en el *nc*. Si el resultado de estas comparaciones es positivo y el emisor del *sc* es confiable para el verificador, el *sc* es legítimo.

Un *Camino de Certificados Anidados* es una cadena de certificados anidados con un certificado clásico al final. Para crear dichos caminos, cada CA expide certificados anidados a todos los certificados que ha expedido su sucesor. Así, un camino de *k* certificados anidados puede verse como se ilustra en la Figura 5.1, donde: nc_k, nc_{k-1}, … , nc_1 son los certificados anidados; cc_0 es un certificado clásico; A_o, $A_1,\ldots,A_k$ son las autoridades que expiden los certificados; y *T* es la entidad objetivo, que quiere verificarse.

Para hacer la verificación, sólo el primer certificado anidado del camino, nc_k, es verificado criptográficamente usando la clave pública de su emisor, A_k. Los otros certificados en el camino son verificados como en el caso de un solo certificado sujeto.

Más tarde, en el año 2000, los mismos autores, proponen NPKI (*Nested certificate based PKI*) [LC00] que es una reforma de la PKI tradicional usando certificados anidados. Aunque este método reduce el número de verificaciones criptográficas, su desventaja es el gran número de certificados anidados que deben expedir las NCAs para formar los caminos en la extensa red de certificados. Esto conlleva a incrementar la complejidad de la infraestructura y dificulta su gestión.

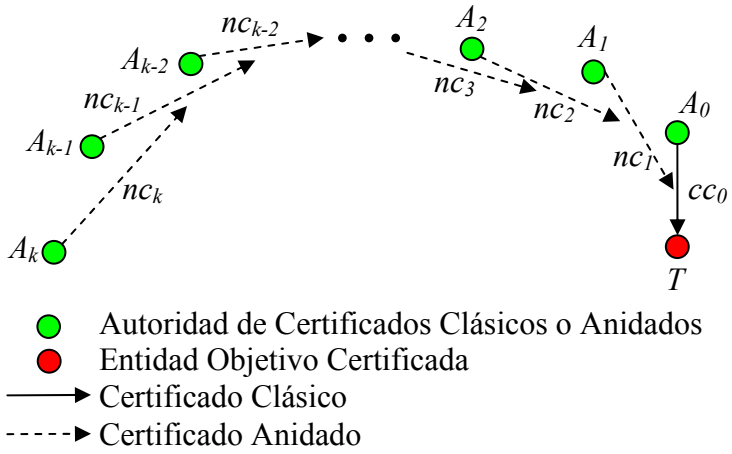

Figura 5.1: Camino de certificados anidados

Nuestra propuesta disminuye el número de operaciones de descifrado con clave pública que debe realizar el verificador durante la validación de caminos, utilizando cadenas de hash para establecer una relación de confianza alternativa entre las entidades de la PKI, lo cual contribuye a disminuir el coste computacional del proceso de validación.

5.3 Cadenas de Hash

Una cadena de hash [Lam81] es una lista de valores $y_1, y_2, ..., y_n$ unidos criptográficamente, donde n es la longitud de la cadena. Estas cadenas se obtienen aplicando repetidamente una función de hash H a una semilla secreta x, como se muestra a continuación.

$$y_1 = H(x)$$
$$y_2 = H(y_1)$$
$$\vdots$$
$$y_n = H(y_{n-1})$$

Las cadenas de hash son la base de nuestra propuesta, de allí que incluyamos su definición.

5.4 Escenario de Estudio

Dos usuarios U y V hacen parte de la misma PKI jerárquica. RCA es la base de confianza de la arquitectura, por tanto, su clave pública PK_{RCA} se da a conocer a todos los usuarios de la PKI en el momento en que se registran en ella. Si el usuario V recibe un mensaje M firmado por U y quiere verificar la firma de dicho mensaje, debe primero validar el camino de certificación de U para obtener su clave pública PK_U. En la Figura 5.2 se puede observar con mayor claridad este escenario.

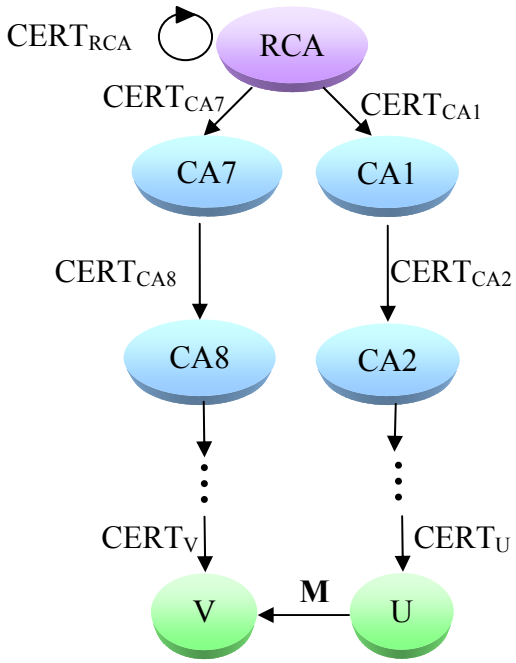

Figura 5.2: Escenario de estudio

Las flechas indican los certificados expedidos de una entidad a otra. Así, RCA expide los certificados $CERT_{CA1}$ y $CERT_{CA7}$ a las autoridades CA1 y CA7 respectivamente; CA1 expide un certificado a CA2 ($CERT_{CA2}$) y así sucesivamente hasta que la autoridad $CA_{Lc\text{-}1}$ le expide un certificado al usuario U ($CERT_U$), donde L_c es la longitud del camino de certificación del usuario U. De manera similar se forma el camino de certificación de V.

El círculo con flecha al lado izquierdo de RCA significa que esta autoridad se expide su propio certificado, es decir, $CERT_{RCA}$ es un certificado auto-firmado [ITU00]. La notación utilizada en este capítulo se encuentra en la Tabla 5.1.

Tabla 5.1: Notación TRUTHC

NOTACIÓN	SIGNIFICADO
PK_X	Clave pública de la entidad X
SK_X	Clave privada de la entidad X
$CERT_X$	Certificado de la entidad X
Cnt_X	Contenido del certificado de la entidad X
Sig_X	Firma del certificado de la entidad X
L_c	Longitud del camino de certificación
n_X	Semilla secreta de la autoridad X
s_{RCA}	Semilla secreta aleatoria de la CA raíz
h_X	Valor de chequeo de integridad de la entidad X
SN_X	Número serial del certificado de la entidad X
N_X	Semilla secreta cifrada de la autoridad X
OP_{hash}	Número de operaciones de hash
T_{hash}	Tiempo de ejecución de una operación de hash
OP_{en}	Número de operaciones de cifrado con clave pública
T_{en}	Tiempo de ejecución de una operación de cifrado con clave pública
OP_{dec}	Número de operaciones de descifrado con clave privada
T_{dec}	Tiempo de ejecución de una operación de descifrado con clave privada
OP_{sig}	Número de operaciones de firma
T_{sig}	Tiempo de ejecución de una operación de firma
OP_{ver}	Número de operaciones de verificación
T_{ver}	Tiempo de ejecución de una operación de verificación

El certificado de una entidad X ($CERT_X$) se compone a su vez de un contenido (Cnt_X) y la firma sobre ese contenido (Sig_X):

$$CERT_X = Cnt_X + Sig_X \quad \textbf{(5.1)}$$

Para obtener la parte firmada de un certificado, su emisor CAi aplica la función de hash *H* sobre el contenido del certificado y luego realiza una operación de cifrado con su clave privada SK_{CAi} sobre el valor de hash resultante:

$$Sig_X = SK_{CAi}\,(H(Cnt_X)) \quad \textbf{(5.2)}$$

Si V quiere verificar el camino de certificación del usuario U - $CERT_{CA1}$, $CERT_{CA2}$,…, $CERT_U$ - debe comprobar primero la firma de $CERT_{CA1}$, para lo cual calcula el hash de Cnt_{CA1} y luego realiza una operación de verificación sobre la parte firmada del mismo certificado (Sig_{CA1}). Para ello utiliza la clave pública de la base de confianza (PK_{RCA}). Si los dos resultados coinciden, se puede confiar en el contenido de $CERT_{CA1}$ y por tanto en la clave pública que contiene (PK_{CA1}). Con esta clave pública se verifica la firma del siguiente certificado en el camino ($CERT_{CA2}$) y así sucesivamente hasta obtener la clave pública del usuario U (PK_U). Por tanto, si la longitud del camino de certificación es L_c, se requerirán L_c operaciones de hash y L_c operaciones de verificación para comprobar la firma de todos los certificados en el camino. La Figura 5.3 muestra la forma en que se relacionan los certificados de un camino de certificación.

Cuando se verifica la firma de los certificados que hacen parte de un camino de certificación se persiguen básicamente dos objetivos:

1. Asegurar la integridad de los certificados que hacen parte del camino. Esto se consigue mediante las operaciones de hash

2. Verificar el origen de los certificados o la relación de confianza que existe entre las diferentes entidades que hacen parte del camino. Esto se hace básicamente a través de las operaciones de verificación.

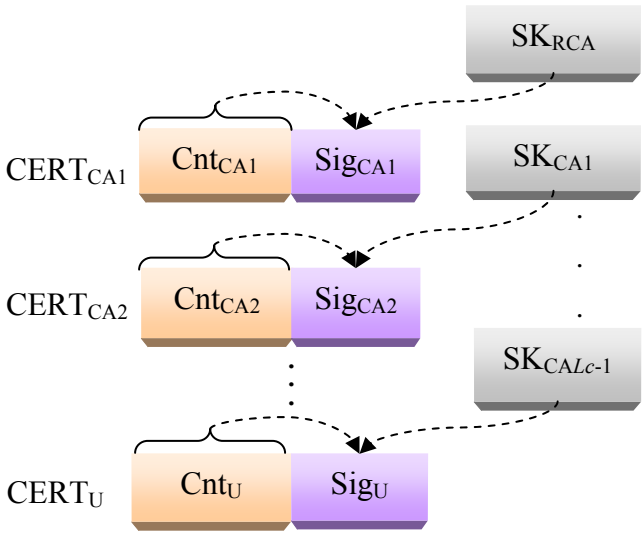

Figura 5.3: Cadena de certificados

Ya que el tiempo requerido para llevar a cabo una operación de verificación es mucho mayor que el tiempo que toma una operación de hash (ver Tablas 3.2 y 3.6), se puede reducir el coste computacional del verificador si se disminuye el número de operaciones de verificación necesarias durante el proceso de validación de caminos. Esto lo podemos hacer estableciendo una relación de confianza diferente entre las entidades que hacen parte de la PKI jerárquica.

La disminución del número de operaciones de verificación contribuirá a la reducción del coste computacional del proceso de validación, que según se concluyó en el capítulo 3 es crítico para dispositivos con capacidad de procesamiento limitada, como teléfonos móviles, tarjetas inteligentes, etc.

5.5 Descripción de TRUTHC

TRUTHC establece una relación de confianza alternativa entre las entidades de una PKI jerárquica a través de dos cadenas de hash: una encadena las semillas secretas de las CAs y la otra encadena los certificados de cada camino. Con esto reemplazamos las operaciones de verificación por operaciones de hash en el proceso de validación de caminos de certificación, lo que reduce el coste computacional del verificador. Por tanto, TRUTHC proporciona un mecanismo alternativo de validación de cadenas de certificados que puede utilizarse cuando el verificador disponga de poca capacidad de procesamiento y sea necesario delegar la función de validación en otra entidad, como sucede con las redes móviles que poseen servidores de validación.

Para describir esta propuesta utilizamos la metodología usada por Karjoth et al, en [KAG98].

5.5.1 Expedición de Certificados

TRUTHC extiende el proceso de expedición de certificados típico, explicado en [ITU00].

El protocolo inicia cuando la RCA elige una semilla secreta aleatoria s_{RCA} y aplica un hash sobre ella para obtener la semilla secreta n_{RCA}.

$$n_{RCA} = H(s_{RCA}) \quad \textbf{(5.3)}$$

Luego, RCA expide el certificado $CERT_{CA1}$ a la autoridad CA1 y utiliza la semilla secreta $n_{RCA,}$ y el contenido de $CERT_{CA1}$ (Cnt_{CA1}), para calcular el valor de chequeo de integridad h_{CA1}. También utiliza n_{RCA} y el número de serie de $CERT_{CA1}$ (SN_{CA1}) para generar la semilla secreta de la autoridad CA1 (n_{CA1}), así:

$$h_{CA1} = H(n_{RCA}, Cnt_{CA1}) \quad \textbf{(5.4)}$$

$$n_{CA1} = H(n_{RCA}, SN_{CA1}) \quad \textbf{(5.5)}$$

Finalmente, RCA le envía a CA1: el certificado de la base de confianza $CERT_{RCA}$, el certificado $CERT_{CA1}$, el valor de chequeo de integridad h_{CA1} y la semilla secreta n_{CA1}, cifrada con la clave pública de CA1 (PK_{CA1}), de manera que sólo CA1 pueda descifrarla.

Más tarde, CA1 obtiene la semilla secreta n_{CA1} realizando una operación de descifrado con su clave privada SK_{CA1} sobre la semilla cifrada que recibió de RCA.

$$n_{CA1} = SK_{CA1}\,(PK_{CA1}(n_{CA1})) \quad \textbf{(5.6)}$$

CA1 le expide entonces un certificado a CA2 y obtiene el valor de chequeo de integridad de esta autoridad h_{CA2} y su semilla secreta n_{CA2} de la siguiente manera:

$$h_{CA2} = H(h_{CA1}, n_{CA1}, Cnt_{CA2}) \quad \textbf{(5.7)}$$

$$n_{CA2} = H(n_{CA1}, SN_{CA2}) \quad \textbf{(5.8)}$$

Luego, CA1 le envía a CA2: $CERT_{RCA}$, $CERT_{CA2}$, h_{CA2} y N_{CA2}, que es la semilla de CA2 cifrada con su clave pública PK_{CA2}.

$$N_{CA2}=PK_{CA2}(n_{CA2}) \quad \textbf{(5.9)}$$

Y así sucesivamente, hasta que el usuario U recibe de su autoridad de certificación CA_{Lc-1} el certificado de la base de confianza $CERT_{RCA}$, su certificado $CERT_U$ y el valor de chequeo de integridad h_U.

Por tanto, se crea una cadena de hash con las semillas secretas (n_{CAi}) y el número de serie de los certificados (SN_{CAi}):

$$n_{RCA} = H(s_{RCA})$$
$$n_{CA1} = H(n_{RCA}, SN_{CA1})$$
$$n_{CA2} = H(n_{CA1}, SN_{CA2})$$
$$\vdots$$
$$n_{CALc-1} = H(n_{CALc-2}, SN_{CA\,Lc-1})$$

Donde: $L_c - 1$ es el número de CAs intermedias en el camino de certificación.

Y otra cadena de hash con los valores de chequeo de integridad (h_{CAi}), las semillas secretas (n_{CAi}) y el contenido de los certificados (Cnt_{CAi}):

$$h_{CA1} = H(n_{RCA}, Cnt_{CA1})$$
$$h_{CA2} = H(h_{CA1}, n_{CA1}, Cnt_{CA2})$$
$$\vdots$$
$$h_{CALc-1} = H(h_{CALc-2}, n_{CALc-2}, Cnt_{CALc-1})$$
$$h_U = H(h_{CALc-1}, n_{CALc-1}, Cnt_U)$$

La relación de encadenamiento de las semillas y de los certificados, el cifrado de las semillas y el protocolo pueden definirse así:

Relación de encadenamiento

$n_{RCA} = H(s_{RCA})$

$n_{CA1} = H(n_{RCA}, SN_{CA1})$

$n_{CAi} = H(n_{CAi-1}, SN_{CAi})$, $2 \le i \le L_c - 1$

$h_{CA1} = H(n_{RCA}, Cnt_{CA1})$

$h_{CAi} = H(h_{CAi-1}, n_{CAi-1}, Cnt_{CAi}), 2 \le i \le L_c - 1$

$h_U = H(h_{CALc-1}, n_{CALc-1}, Cnt_U)$

Semilla Cifrada

$N_{CAi} = PK_{CAi}(n_{CAi}), 1 \le i \le L_c - 1$

Protocolo

$CAi \rightarrow CAi+1: CERT_{RCA}, CERT_{CAi+1}, h_{CAi+1}, N_{CAi+1}$

$CA_{Lc-1} \rightarrow U : CERT_{RCA}, CERT_U, h_U$

Se asume que la función de hash *H* es libre de colisión.

La Figura 5.4 muestra la relación que se establece ahora entre los certificados utilizando cadenas de hash.

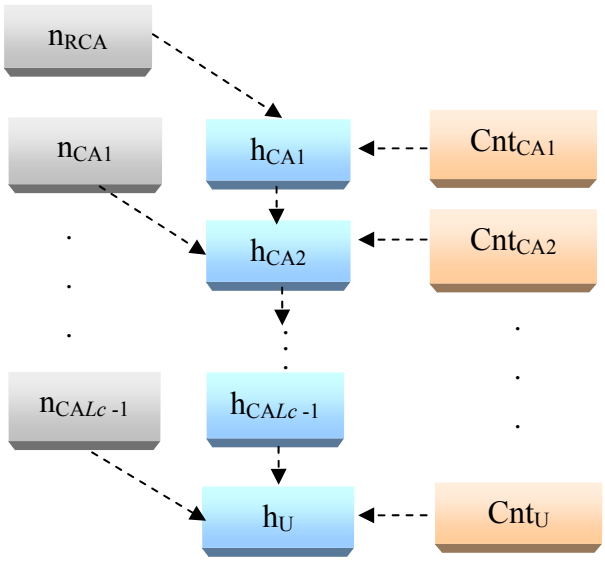

Figura 5.4: Encadenamiento de certificados mediante cadenas de hash

5.5.2 Verificación de Certificados

Para verificar los caminos de certificación que se forman con TRUTHC, adicionamos una tercera parte de confianza a la PKI jerárquica llamada Autoridad de Verificación (VA) (Figura 5.5).

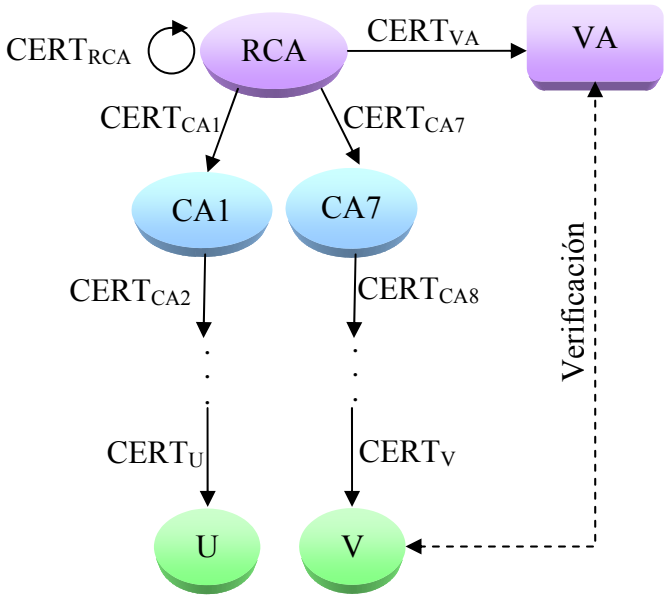

Figura 5.5: Modelo de verificación

VA se encarga de verificar la integridad de los certificados y la relación de confianza entre las entidades que hacen parte del camino de certificación.

RCA emite el certificado de la VA ($CERT_{VA}$) y le envía: el certificado de la base de confianza $CERT_{RCA}$, el certificado $CERT_{VA}$ y la semilla n_{RCA} cifrada con la clave pública PK_{VA} para asegurar su confidencialidad.

Semilla Cifrada

$N_{VA} = PK_{VA}(n_{RCA})$

Protocolo

$CA \rightarrow VA$: $CERT_{RCA}$, $CERT_{VA}$, N_{VA}

Para verificar el camino de certificación del usuario U:

1. El usuario V obtiene h_U y el certificado $CERT_U$.
2. V envía a VA el certificado $CERT_U$
3. VA recupera los demás certificados en el camino de certificación de U: $CERT_{CA1}$, $CERT_{CA2}$,..., $CERT_{Lc\text{-}1}$.
4. VA calcula h'_U, empleando las ecuaciones de relación de encadenamiento especificadas en la sección 5.5.1, la semilla secreta n_{RCA}, el certificado $CERT_U$ y los demás certificados del camino de certificación de U.
5. VA envía el valor de chequeo de integridad h'_U al usuario V en una respuesta firmada.
6. El usuario V verifica la firma de la respuesta de VA. Esto implica verificar la firma de $CERT_{VA}$ con la clave pública PK_{RCA} para obtener PK_{VA} y verificar luego la firma de la respuesta.
7. Si h_U y h'_U son iguales, V habrá comprobado la integridad del certificado $CERT_U$ y su pertenencia al camino de certificación de U.

Las funciones de la autoridad de verificación pueden estar integradas en la autoridad de certificación raíz o pueden distribuirse entre las diferentes autoridades de la jerarquía. Ya que una autoridad puede calcular las semillas de sus CAs subordinadas. El verificador sólo tendría que buscar a la primera autoridad que tenga en común con la entidad objetivo y encomendarle a ésta la labor de verificación.

5.6 Integración de TRUTHC con los Certificados X.509

El valor de chequeo de integridad h_{CAi} puede incluirse como una extensión más en los certificados X.509, de manera que TRUTHC es compatible con ellos. Sin embargo, cuando la VA vaya a calcular el hash sobre el contenido de algún certificado (Cnt_{CAi}), deberá excluir esta extensión para que el valor de h'_U sea correcto.

Adicionar esta extensión a un certificado X.509 implicará un leve incremento en su tamaño, por ejemplo, si se usa SHA-1 como función de hash, el tamaño de los valores de hash será únicamente 20 bytes.

Es importante aclarar, que TRUTHC no pretende sustituir el método tradicional de verificación de firma de los certificados, sólo es un método alternativo que puede aplicarse cuando los verificadores poseen poca capacidad de procesamiento.

5.7 Análisis de Seguridad

TRUTHC cuenta con las siguientes características de seguridad:

- ***Confidencialidad de las semillas:*** Gracias a las propiedades de la criptografía de clave pública, una semilla cifrada $N_{CAi}=PK_{CAi}(n_{CAi})$ sólo puede ser descifrada con la clave privada de la autoridad CAi. De esta manera, cada CA de la PKI conocerá su propia semilla y podrá calcular las semillas de sus CAs subordinadas, pero no podrá conocer la semilla de las autoridades superiores en la jerarquía.
- ***Verificabilidad de la integridad de los certificados:*** Sólo las CAs pueden calcular los valores de chequeo de integridad h_{CAi} de los certificados que expiden gracias a que son las únicas que conocen las semillas secretas n_{CAi}. Por tanto, si el valor h_U recuperado por un verificador V coincide con el valor h'_U que le retorna VA, este verificador puede confiar en la integridad de los certificados recuperados, y en que todos hacen parte del mismo camino de certificación.
- ***Integridad de los certificados:*** Si un atacante modifica el contenido de un certificado $CERT_{CAi}$ y deja intacto el valor h_{CAi}, para que se mantenga la relación de encadenamiento con el contenido modificado Cnt'_{CAi} se debe cumplir:

$$H(h_{CAi-1},n_{CAi-1},Cnt'_{CAi})=H(h_{CAi-1},n_{CAi-1},Cnt_{CAi})$$

Pero esto no cumple con la condición que la función de hash *H* sea libre de colisión. Por tanto, no es posible modificar el contenido de un certificado en el

camino de certificación sin modificar la relación de encadenamiento que existe entre ellos.

- ***Resistencia a la inserción:*** Si el atacante quiere incluir un nuevo certificado en el camino y el esquema de cifrado es seguro, aunque disponga de h_{CAi}, no podrá obtener el valor de n_{CAi}, necesario para calcular h_{CAi+1}.
- ***Verificabilidad de la integridad de la cadena de hash:*** Si el atacante modifica el valor de algún h_{CAi} para propiciar un ataque de negación de servicio (DoS), cuando la VA encuentre un h_{CAi} erróneo puede verificar la firma del certificado implicado para asegurar la integridad de dicho valor de la cadena.

5.8 Evaluación

En esta sección se compara el coste computacional de los procesos de expedición de certificados y verificación de firma de una PKI típica y una PKI con TRUTHC. Los cálculos se basan en el número de operaciones criptográficas necesarias para realizar dichos procesos. Para ello se utiliza SHA-1 [NIS95] como función de hash y RSA-1024 [RSA78]como algoritmo de clave pública.

Como tiempos de ejecución de las operaciones criptográficas realizadas por las CAs y la VA se toman los especificados en la Tabla 5.2, que son de un ordenador con procesador Pentium 4 a 2,1GHz y sistema operativo Windows XP SP1 [Dai04]. Y como tiempos de estas operaciones para un verificador con terminal móvil, se toman los mostrados en la Tabla 5.3. Estos son valores obtenidos de una PDA Compaq iPAQ H3630 con procesador StrongARM a 206 MHz y sistema operativo Windows CE Pocket PC 2002 [AVT04].

Tabla 5.2: Tiempo de ejecución de operaciones criptográficas de CAs y VA

ALGORITMO	TIEMPO DE EJECUCIÓN
SHA-1	14,03 ns/byte
RSA-1024 Cifrado	0,18ms/operación
RSA-1024 Descifrado	4,77ms/operación
RSA-1024 Firma	4,75ms/operación
RSA-1024 Verificación	0,18ms/operación

Tabla 5.3: Tiempo de ejecución de operaciones criptográficas del verificador

ALGORITMO	TIEMPO DE EJECUCIÓN
SHA-1	0,19 ms/operación
RSA-1024 Firma	78,25 ms/operación
RSA-1024 Verificación	5,01 ms/operación

Se considera que los certificados de los usuarios son como el certificado cliente del ejemplo D.1 en [WAP01g] que ocupa 425 bytes, de los cuales 270 bytes corresponden a su contenido; y los certificados de las CAs y la VA como el certificado de CA del ejemplo D.2 en [WAP01g], que ocupa 473 bytes, de los cuales 318 bytes corresponden a su contenido. Además, como se utiliza SHA-1, el tamaño de cada valor de hash va a ser 20 bytes (160bits). Se considera igualmente que el tamaño de la semilla secreta aleatoria s_{RCA} es 20 bytes.

Para calcular el coste computacional se utiliza la ecuación (5.10) que es una extensión de la ecuación (3.1).

$$COST = (OP_{hash}*T_{hash})+(OP_{en}*T_{en})+(OP_{dec}*T_{dec})+(OP_{sig}*T_{sig})+(OP_{ver}*T_{ver}) \qquad \textbf{(5.10)}$$

5.8.1 Expedición de Certificados

5.8.1.1 PKI Típica

Cuando una CA expide un certificado realiza un hash sobre el contenido del certificado (Cnt_{CAi}) y luego una operación de firma sobre ese hash. Si el certificado es como el del ejemplo D.2 en [WAP01g], cuyo contenido ocupa 318 bytes, el coste computacional del proceso de expedición de ese certificado para las CAs de la PKI, utilizando los tiempos de la Tabla 5.2 y la ecuación (5.10), será (ver Tabla 5.4):

$$COST = (1*14{,}03*10^{-9}*318) + (1*4{,}75*10^{-3}) = \textbf{4,75ms}$$

5.8.1.2 PKI con TRUTHC

Aquí, el coste del proceso de expedición de certificados para la autoridad de certificación raíz es diferente que el de una CA subordinada.

Cuando una CA subordinada expide un certificado a otra CA, realiza las siguientes operaciones:

- Una operación de hash y una de firma
  $Sig_{CAi+1} = SK_{CAi}(H(Cnt_{CAi+1}))$
- Una operación de descifrado con clave privada :
  $n_{CAi} = SK_{CAi}(N_{CAi})$
- Dos operaciones de hash
  $h_{CAi+1} = H(h_{CAi}, n_{CAi}, Cnt_{CAi+1})$
  $n_{CAi+1} = H(n_{CAi}, SN_{CAi+1})$
- Una operación de cifrado con clave pública
  $N_{CAi+1} = PK_{CAi+1}(n_{CAi+1})$

Por tanto, el coste computacional de expedición de un certificado como el del ejemplo D.2 en [WAP01g] por parte de una CA subordinada será (ver Tabla 5.4):

$$COST = (318 + 358 + 21) * 14{,}03 * 10^{-9} + 0{,}18 * 10^{-3} + 4{,}77 * 10^{-3} + 4{,}75 * 10^{-3} = \mathbf{9{,}71ms}$$

RCA en cambio no tiene que descifrar su semilla pues ella misma la genera, realizando una operación de hash sobre la semilla secreta aleatoria en lugar de una operación de descifrado con clave privada. Las operaciones criptográficas que lleva a cabo RCA son entonces:

- Una operación de hash y una de firma
  $Sig_{CAi} = SK_{RCA}(H(Cnt_{CAi}))$
- Tres operaciones de hash
  $n_{RCA} = H(s_{RCA})$
  $h_{CAi} = H(n_{RCA}, Cnt_{CAi})$
  $n_{CAi} = H(n_{RCA}, SN_{CAi})$
- Una operación de cifrado

$N_{CAi} = PK_{CAi}(n_{CAi})$

Así, el coste de expedición de un certificado como el del ejemplo D.2 en [WAP01g] por parte de RCA será (ver Tabla 5.4):

$$COST = (318 + 20 + 338 + 21) * 14{,}03*10^{-9} + 0{,}18*10^{-3} + 4{,}75*10^{-3} = \mathbf{4{,}94ms}$$

Tabla 5.4: Coste computacional del proceso de expedición de certificados

CASO	EMISOR	COSTE COMPUTACIONAL
PKI Típica	RCA, CA	4,75ms
TRUTHC	CA	9,71ms
TRUTHC	RCA	4,94ms

5.8.2 Verificación de Certificados

5.8.2.1 PKI Típica

Cuando un verificador comprueba la firma de todos los certificados en un camino de certificación realiza en total L_c operaciones de hash y L_c operaciones de verificación. El coste computacional de este proceso, utilizando los datos de la Tabla 5.3, es:

$$COST = (L_c*0{,}19*10^{-3})+(L_c*5{,}01*10^{-3}) = 5{,}20*10^{-3}*L_c \qquad \mathbf{(5.11)}$$

5.8.2.2 PKI con TRUTHC

En este caso, el verificador chequea la firma de la respuesta que le envía VA. Para ello debe verificar la firma de $CERT_{VA}$ con la clave pública de RCA (PK_{RCA}) y luego con la clave pública de VA (PK_{VA}) verificar la firma de la respuesta. Por tanto, el verificador realiza dos operaciones de hash y dos operaciones de verificación, cuyo coste computacional es:

$$COST= (2*0{,}19*10^{-3}) + (2*5{,}01*10^{-3}) = \mathbf{10{,}40ms}$$

Por otra parte, VA debe realizar:

- Una operación de descifrado con clave privada
  $n_{RCA} = SK_{VA}(PK_{VA}(n_{RCA}))$
- $L_c - 1$ operaciones de hash
  $n_{CA1} = H(n_{RCA}, SN_{CA1})$
  $n_{CAi} = H(n_{CAi-1}, SN_{CAi}), 2 \leq i \leq L_c\text{-}1$
- L_c operaciones de hash
  $h_{CA1} = H(n_{RCA}, Cnt_{CA1})$
  $h_{CAi} = H(h_{CAi-1}, n_{CAi-1}, Cnt_{CAi}), 2 \leq i \leq L_c\text{-}1$
  $h'_U = H(h_{CALc\text{-}1}, n_{CALc\text{-}1}, Cnt_U)$
- Una operación de hash y una de firma
  $Sig_{VAresp} = SK_{VA}(H(Cnt_{VAresp}))$

Donde: VAresp es la respuesta firmada de VA. Por motivos de simplicidad, suponemos que esta respuesta sólo contiene h'_U.

El coste computacional de la VA sería:

$$\begin{aligned} COST &= ((L_c\text{-}1)*21 + 338 + (L_c\text{-}2)*358 + 310+20)*14{,}03*10^{-9} + 4{,}77*10^{-3} + 4{,}75*10^{-3} \\ &= 5{,}32*10^{-6}*L_c + 9{,}52*10^{-3} \end{aligned} \quad \textbf{(5.12)}$$

La Figura 5.6 compara el coste computacional del verificador cuando lleva a cabo la verificación de firma de los certificados en una PKI típica y el coste computacional de la VA, en una PKI con TRUTHC para diferentes valores de L_c. La tabla de datos de esta figura se encuentra en el apéndice A, Tabla A.11.

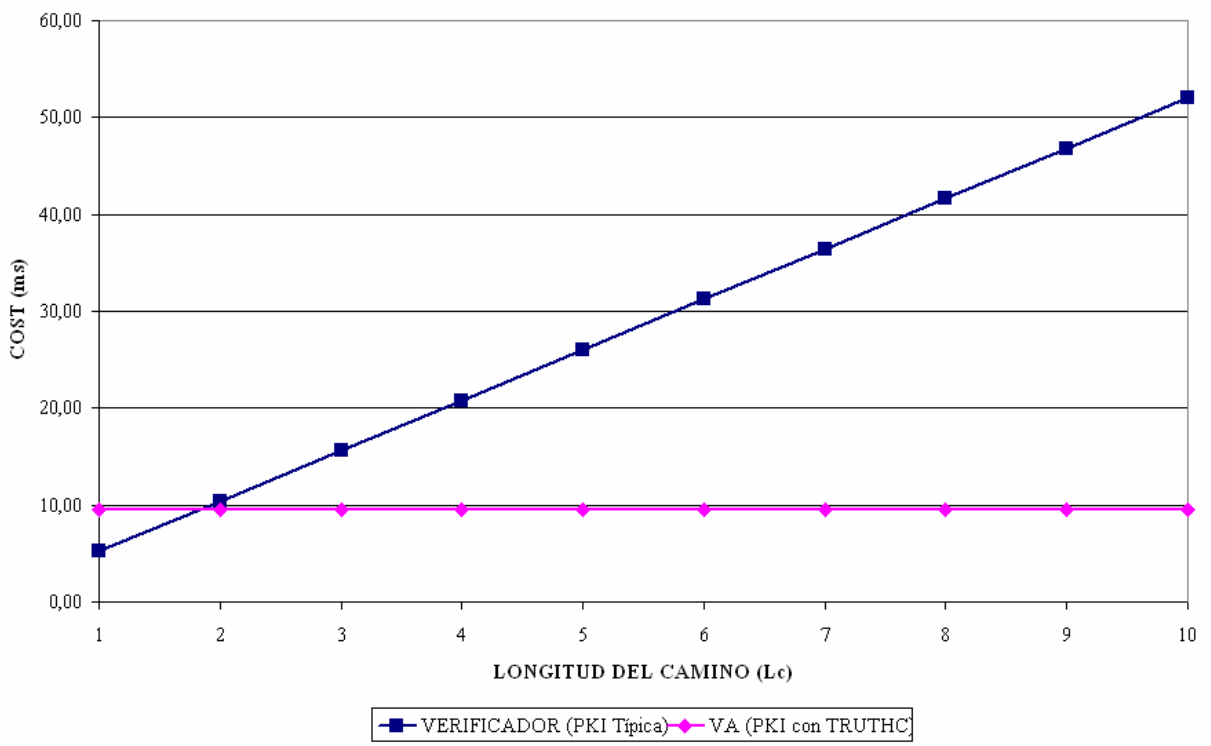

Figura 5.6: Coste computacional de TRUTHC vs. PKI típica

5.9 TRUTHC en Validación de Arquitecturas de Autorización

Las arquitecturas de autorización basadas en certificados pueden utilizar TRUTHC como parte de su servicio de autenticación. Además, si estas arquitecturas implementan algún tipo de delegación de privilegios, TRUTHC podría adaptarse para verificar los caminos de delegación resultantes. En el caso de PMI, por ejemplo, la SOA haría el mismo papel que la RCA en la PKI: elegiría una semilla secreta aleatoria (s_{SOA}) y usando los números seriales de los certificados de atributos se generarían las semillas de las AAs. Por otra parte, los valores de chequeo de integridad se obtendrían a partir del contenido de los certificados de atributos y de las semillas de las autoridades. A continuación se muestra la relación de encadenamiento, la semilla cifrada y el protocolo que se llevaría a cabo en PMI.

Relación de encadenamiento

$n_{SOA} = H(s_{SOA})$

$n_{AA1} = H(n_{SOA}, SN_{AA1})$

$n_{AAi} = H(n_{AAi-1}, SN_{AAi}), 2 \leq i \leq L_d - 1$

$h_{AA1} = H(n_{SOA}, Cnt_{AA1})$

$h_{AAi} = H(h_{AAi-1}, n_{AAi-1}, Cnt_{AAi}), 2 \leq i \leq L_d - 1$

$h_U = H(h_{AALd-1}, n_{AALd-1}, Cnt_U)$

Semilla Cifrada

$N_{AAi} = PK_{AAi}(n_{AAi}), 1 \leq i \leq L_d - 1$

Protocolo

$AAi \rightarrow AAi+1$: $CERT_{SOA}, CERT_{AAi+1}, h_{AAi+1}, N_{AAi+1}$

$AA_{Ld-1} \rightarrow U$: $CERT_{SOA}, CERT_U, h_U$

Los valores de chequeo de integridad serían incluidos como extensión de los certificados de atributos, y el verificador de privilegios desempeñaría la labor de la VA.

5.10 Conclusiones

En este capítulo proponemos TRUTHC, un método alternativo de verificación de cadenas de certificados que reduce el coste computacional de las operaciones criptográficas realizadas por el verificador. TRUTHC establece la relación de confianza entre las entidades de una PKI jerárquica a través de dos cadenas de hash: una que encadena las semillas secretas de las CAs y otra que encadena el contenido de los certificados de cada camino de certificación. Gracias a esta relación de confianza alternativa se reduce el número de operaciones de verificación durante el proceso de validación de caminos, permitiendo verificar la integridad de los certificados y constatar que cada uno de ellos pertenece al mismo camino mediante operaciones de hash. Estas operaciones de hash son realizadas por una autoridad de

verificación y su coste computacional es más bajo que el coste del verificador en una PKI típica, como se muestra en la Figura 5.6. El coste de la VA aumenta muy levemente a medida que se incrementa la longitud del camino de certificación. Por otra parte, el coste computacional del verificador en TRUTHC es constante y puede reducirse a la mitad si el verificador conoce la clave pública de la VA desde el momento en que entra a formar parte de la PKI.

El coste del proceso de expedición de certificados por parte de la RCA es similar para una PKI típica y una PKI con TRUTHC, pero es mayor para las CAs subordinadas cuando se utiliza TRUTHC (ver Tabla 5.4). Sin embargo, el coste de expedición de un certificado no es muy significativo, gracias a la gran capacidad de procesamiento de las CAs.

Una de las ventajas de este método es su compatibilidad con los certificados X.509, ya que los valores de chequeo de integridad h_{CAi} pueden incorporarse en estos certificados como una extensión.

La seguridad del método propuesto depende en gran medida de la confidencialidad de las semillas n_{CAi}, que sólo deben ser conocidas por las CAs y la VA, por lo que deben ser guardadas de manera segura, como si se tratase de la clave privada de la autoridad. El compromiso de una de estas semillas ocasionaría la revocación del certificado de la autoridad propietaria de la semilla y de los certificados expedidos por ella.

TRUTHC no pretende sustituir el método tradicional de verificación de firma de los certificados, y su aplicabilidad se reduce a entornos donde los dispositivos tengan limitada capacidad de procesamiento y deba delegarse el proceso de validación a otra entidad, como en el caso de las redes móviles que disponen de servidores de validación.

El método propuesto puede emplearse únicamente en entornos de autenticación y autorización donde las entidades estén jerárquicamente relacionadas entre ellas.

Capítulo 6

PROSEARCH (Protocol to Simplify the Certification Path Discovery Constructing a Hierarchy)

6.1 Introducción 119
6.2 Validación de Caminos en Arquitecturas Descentralizadas 120
6.3 Descripción de PROSEARCH 121
6.4 Análisis de Seguridad 126
6.5 Ejemplo de Aplicación 127
6.6 PROSEARCH en Arquitecturas de Autorización 140
6.7 Conclusiones 140

6.1 Introducción

Mientras el capítulo 5 se centró en reducir el número de operaciones de verificación realizadas durante el proceso de validación de caminos, este capítulo se centra en otra parte importante de dicho proceso, como es la construcción de los caminos de certificación.

Cuando las relaciones de confianza son unidireccionales, los caminos están bien definidos y son fáciles de encontrar. Sin embargo, si las relaciones de confianza son bidireccionales es difícil construirlos puesto que no todas las opciones conducen a la entidad objetivo y pueden existir múltiples caminos entre dos entidades. Esto dificulta la labor del verificador e incrementa la duración del proceso de validación.

En [SPF06a], [SFH06]y [SPF06c], presentamos el protocolo PROSEARCH (*PROtocol to Simplify the cErtification pAth discoveRy Constructing a Hierarch*y), que establece un modelo de confianza jerárquico a partir de una PKI con relaciones de confianza bidireccionales, basándose en la confiabilidad de las autoridades participantes. Éste permite simplificar el descubrimiento de los caminos de certificación puesto que en la arquitectura jerárquica existe un solo camino entre dos entidades.

El contenido de los artículos mencionados en el párrafo anterior se expresa a lo largo del presente capítulo. La sección 6.2 explica diferentes propuestas que incrementan la eficiencia del proceso de validación en arquitecturas descentralizadas usando un modelo jerárquico. En la sección 6.3 describimos el funcionamiento de nuestro protocolo. La sección 6.4 presenta un análisis de seguridad de PROSEARCH. En la sección 6.5 ilustramos el funcionamiento de nuestro protocolo a través de un ejemplo. La sección 6.6 muestra como puede relacionarse PROSEARCH con las arquitecturas de autorización. Finalmente, la sección 6.7 concluye.

6.2 Validación de Caminos en Arquitecturas Descentralizadas

Existen diferentes propuestas que incrementan la eficiencia del descubrimiento de caminos y el proceso de validación en arquitecturas descentralizadas usando un modelo jerárquico. Marchesini y Smith [MS02], por ejemplo, proponen la creación de una arquitectura jerárquica virtual a partir de una red igual a igual (*peer-to-peer*), que permite verificar los caminos de certificación de manera eficiente. Los secretos (claves privadas) se dividen en múltiples partes, de manera que el compromiso de una de ellas no perjudica a toda la arquitectura. Así, los nodos resultantes son CAs virtuales, formados por varias autoridades que comparten una porción de la clave privada y que por tanto actúan de manera colectiva.

Por otra parte, Pan et al. [PLZ05] proponen un esquema que combina diferentes PKIs, usando también una arquitectura jerárquica. Este esquema selecciona una CA raíz y las autoridades restantes son CAs subordinadas que deben solicitar a la CA raíz un nuevo certificado. Los nuevos certificados, sin embargo, usan las claves públicas que poseían previamente las autoridades. Así, el proceso de combinación es rápido y de bajo coste, y el

procesamiento de caminos de certificación es mucho más simple y eficiente que usando certificación cruzada. Desafortunadamente, el proceso de selección de la CA raíz no se define claramente.

A diferencia de estos trabajos, nuestro protocolo no requiere la generación de nuevos certificados para establecer la jerarquía entre las entidades de la PKI, ya que utiliza las relaciones de confianza existentes. Además, PROSEARCH fija un límite máximo en la longitud de los caminos que lo hace adaptable a las características de los usuarios cuyos terminales tienen capacidad limitada de almacenamiento y procesamiento.

6.3 Descripción de PROSEARCH

En esta sección describimos el funcionamiento de PROSEARCH. Este protocolo establece una jerarquía virtual en una PKI con relaciones de confianza bidireccionales, basado en la confiabilidad de las entidades participantes.

Nuestro protocolo mejora la eficiencia del proceso de descubrimiento de caminos de certificación, gracias a que las relaciones de confianza son unidireccionales en una arquitectura jerárquica.

Ya que la validación de caminos largos es difícil para verificadores con capacidades limitadas, la jerarquía es construida considerando una longitud máxima de caminos, cuyo valor puede fijarse teniendo en cuenta las características de los terminales que poseen los usuarios.

PROSEARCH construye la jerarquía de las hojas a la raíz (hacia arriba). Teóricamente, la autoridad más confiable de la arquitectura se convierte en la raíz de la jerarquía, pero en realidad no es siempre posible establecer a priori qué autoridad se convertirá en raíz, ya que cada una de ellas sólo recibe información de sus vecinas y la jerarquía puede llegar a tener más de una raíz. Por tanto, si la jerarquía fuera construida de la raíz a las hojas (hacia abajo) se necesitaría un gran número de mensajes entre las autoridades de la PKI para determinar desde el principio del protocolo qué autoridad es la raíz. Por esta razón, construimos la

jerarquía hacia arriba. Así, las autoridades menos confiables eligen primero una CA superior y las autoridades más confiables tienden a convertirse en CAs superiores.

Algunos aspectos de nuestro protocolo fueron inspirados en el algoritmo propuesto por J. Hernández-Serrano et al. en [HPS05], aunque el área de aplicación es diferente. La Tabla 6.1 muestra la notación utilizada en este capítulo.

Tabla 6.1: Notación PROSEARCH

NOTACIÓN	SIGNIFICADO
L_{MAX}	Longitud máxima permitida de los caminos de certificación
CA_i	Autoridad de certificación i
L_i	Número de certificados de las hojas a la autoridad i
IN_i	Número de autoridades en que CA_i confía (certificados recibidos)
OUT_i	Número de autoridades que confían en CA_i (certificados emitidos)
CA_0	Autoridad actual

Para su mejor comprensión, hemos dividido nuestro protocolo en dos fases:

- ***Orden de confiabilidad entre las CAs****:* En esta fase, las autoridades vecinas son ordenadas de la menos confiable a la más confiable.
- ***Creación de la jerarquía****:* En esta fase se establece una relación de confianza jerárquica entre las CAs de la PKI.

6.3.1 Orden de Confiabilidad entre las CAs

El protocolo inicia cuando una autoridad CA_0 manifiesta a sus CAs vecinas (CAs que le expidieron un certificado a CA_0 y CAs a las que CA_0 emitió un certificado) que desea establecer una jerarquía con ellas. CA_0 además propone una longitud máxima de los caminos de certificación a establecer (L_{MAX}), basada en la capacidad de procesamiento y almacenamiento de sus usuarios. CA_0 envía, por tanto, un mensaje de solicitud a sus vecinos conteniendo el valor de L_{MAX}. Este mensaje y todos los mensajes enviados durante el

protocolo son firmados por la autoridad que los emite y deben ser autenticados por el receptor.

Cada CA vecina tiene la opción de negarse a colaborar en el establecimiento de la jerarquía, enviando un mensaje de rechazo a la autoridad que se lo solicitó, o puede enviar a la CA solicitante un mensaje de aceptación y a sus otras vecinas un mensaje de solicitud conteniendo la longitud máxima establecida L_{MAX}.

Entre las posibles causas para el rechazo de una solicitud están: el no estar de acuerdo con la longitud máxima de los caminos, la poca correspondencia entre las políticas de las dos autoridades, etc.

Una vez la autoridad CA_0 recibe respuesta de todas sus vecinas, establece cuales de ellas están dispuestas a hacer parte de la jerarquía. Así, puede determinar el número de autoridades que le expidieron un certificado y están dispuestas a participar en la jerarquía (IN_0) y el número de autoridades a las que emitió un certificado dispuestas también a participar (OUT_0). CA_0 comparte estos valores con sus vecinas participantes enviándoles un mensaje de información. Estas vecinas le envían también sus propios valores IN_i y OUT_i utilizando el mismo tipo de mensajes.

Seguidamente, CA_0 compara OUT_0 con los OUT_i que recibe de sus vecinas y coloca estos valores en orden del más bajo al más alto. La autoridad con el OUT_i más bajo es la *menos confiable*, es decir, la vecina en la que menos confían las demás participantes. Si existen dos o más autoridades vecinas con el mismo OUT_i, se ordenan de acuerdo a su valor de IN_i, del más bajo al más alto otra vez. Por motivos de simplicidad, no hemos considerado más parámetros para establecer en orden las autoridades, como la correspondencia de políticas o distancia entre ellas, pero estos pueden ser tenidos en cuenta en caso que OUT_i e IN_i sean iguales para dos o más autoridades. En estos casos, nosotros utilizamos el identificador de cada autoridad para ordenarlas.

De esta manera, cada autoridad ordena sus vecinas de la menos confiable a la más confiable, determinando cuales de ellas son menos confiables y más confiables que si misma.

Por otra parte, cuando el protocolo comienza, L_i=0 para todas las autoridades.

El algoritmo que describe el procedimiento llevado a cabo por cada autoridad en la primera fase del protocolo se encuentra en el apéndice B.

6.3.2 Creación de la Jerarquía

En esta fase del protocolo, las autoridades actúan de la menos confiable a la más confiable de acuerdo con el orden establecido en la primera fase. Por tanto, la autoridad menos confiable del vecindario es la primera en actuar mientras las otras esperan la intervención de sus vecinas menos confiables para hacerlo.

El objetivo de la segunda fase es que cada autoridad elija una CA superior entre las vecinas participantes que le emitieron un certificado (vecinas en que confía). Por tanto, cuando la autoridad CA_0 actúa, determina cual de estas vecinas es la más confiable, utilizando el orden establecido en la primera fase del protocolo, y la elige como CA superior. Si L_0 es mayor que el L_i de la CA superior y (L_0+1) es menor o igual a (L_{MAX}-1), el L_i de la CA superior toma el valor (L_0+1). En caso que (L_0+1) sea mayor que (L_{MAX}-1), CA_0 debe cambiar de CA superior y elegir a su siguiente vecina más confiable, siempre y cuando la confiabilidad de esta autoridad sea mayor que la de CA_0. Por supuesto, CA_0 debe verificar de nuevo si L_0 es mayor que el L_i de la nueva CA superior y así sucesivamente hasta que CA_0 encuentre una CA superior apropiada. Es posible que ninguna de las CAs en que confía CA_0, más confiables que ella, cumpla con las condiciones necesarias para convertirse en su CA superior. Por tanto, una vez concluido este procedimiento CA_0 envía un mensaje de asociación a sus vecinas participantes informando la identidad de su CA superior o un mensaje de fallo indicando que no le fue posible elegir una CA superior.

Posteriormente, la siguiente autoridad menos confiable del vecindario, según el orden establecido en la primera fase del protocolo, elegirá su CA superior y así sucesivamente hasta que todas las CAs participantes elijan una.

La autoridad más confiable no debe llevar a cabo esta fase del protocolo porque no existe ninguna CA con mayor grado de confiabilidad que ella. Así, cuando es su turno de actuar, esta

autoridad envía un mensaje de CA raíz a sus vecinas, informándoles que es la raíz de la jerarquía.

Las autoridades que no eligieron una CA superior en la segunda fase del protocolo, y enviaron un mensaje de fallo, también son consideradas CAs raíz. Por tanto, si existe más de una CA raíz al final de la segunda fase, el protocolo debe ser repetido con las raíces resultantes, considerando sólo los certificados expedidos entre ellas para determinar el nuevo valor de los parámetros OUT_i e IN_i. Por ende, cuando la autoridad más confiable de un vecindario recibe un mensaje de fallo, interpreta que debe de reiniciar el protocolo después de enviar el mensaje de CA raíz a sus vecinas.

El L_i de cada CA raíz conserva el valor que adquirió durante la segunda fase del protocolo. Además, cuando se repite PROSEARCH, el valor de L_i puede ser menor o igual a L_{MAX} durante la segunda fase, en lugar de a ($L_{MAX} - 1$) como en la primera ejecución.

Aún así, la jerarquía puede tener más de una CA raíz después de repetir el protocolo. En este caso, las raíces deben encontrar el camino más corto entre ellas utilizando algún método alternativo.

Finalmente, la CA raíz envía su certificado de clave pública a las autoridades que se encuentran bajo ella en la jerarquía a través de un mensaje de certificado raíz.

El algoritmo que describe el procedimiento llevado a cabo por cada autoridad en la segunda fase del protocolo se encuentra en el apéndice B.

6.3.3 Vinculación de una Nueva CA a la Jerarquía

Cuando una nueva autoridad quiera darse de alta en la jerarquía establecida, esta autoridad debe preguntar a una de sus CAs vecinas, que haga parte de la jerarquía, el valor de L_{MAX}. Si está de acuerdo con este valor, debe iniciar el protocolo enviando a sus vecinas un mensaje de solicitud.

Las modificaciones que sufra la jerarquía van a depender de qué tan confiable sea la nueva autoridad, así, si esta CA es la menos confiable de su vecindario, las demás autoridades no modificarán su CA superior.

6.3.4 Salida de una CA de la Jerarquía

Cuando una autoridad perteneciente a la jerarquía salga de ella, sus vecinas participantes deben iniciar de nuevo el protocolo para establecer una nueva jerarquía.

En este caso, las modificaciones que sufra la jerarquía también dependerán del nivel de confiabilidad de la autoridad que sale de la jerarquía, pues si ésta no es la CA superior de ninguna autoridad, las demás no modificarán su CA superior. Por el contrario, cuando la autoridad que abandona la jerarquía es una CA superior, sus CAs subordinadas deben llevar a cabo el protocolo con sus vecinas.

6.4 Análisis de Seguridad

PROSEARCH garantiza la integridad y autenticidad de los mensajes que se comparten durante el protocolo gracias a que estos son firmados digitalmente por las autoridades que los emiten. Por tanto, verificando su firma digital se puede comprobar que el mensaje no ha sido modificado en su camino al receptor y que la identidad de la autoridad que expidió dicho mensaje es auténtica.

Aunque las entidades que llevan a cabo el protocolo son terceras partes de confianza y teóricamente se puede confiar en la veracidad de la información que envían, si alguna de ellas fuera una entidad maliciosa podría tratar de perpetrar un ataque de negación del servicio al inicio del protocolo, en el momento en que se determina qué autoridades van a participar en la formación de la jerarquía. Este ataque podría consistir en enviar un valor de L_{MAX} muy elevado a sus vecinas de tal manera que estas se nieguen a participar en la jerarquía. Sin embargo, estas autoridades podrían recibir el valor real de L_{MAX} por parte de alguna de sus otras vecinas en un mensaje de solicitud, lo que les permitiría detectar que la información recibida anteriormente es falsa y que esa autoridad es maliciosa.

Otro tipo de ataque que podría llevar a cabo una entidad maliciosa es incrementar el valor de sus parámetros IN_i y OUT_i para tratar de convertirse en la raíz de la jerarquía. Sin embargo esto no siempre es posible, como se ejemplifica en el ejemplo de aplicación, sección 6.5.8, ya

que la jerarquía resultante va a depender de las relaciones de confianza existentes entre las CAs. Por tanto, no siempre la CA raíz va a ser la que tenga un mayor valor de IN_i y OUT_i, aunque esto sea lo más probable. Adicionalmente, cada autoridad elige como CA superior a la que considere más confiable. Por tanto, aunque los valores de IN_i y OUT_i son un indicativo de las relaciones de confianza existentes, para una autoridad en particular pueden ser más importantes otros parámetros como la correspondencia entre políticas o la distancia a entre las autoridades, valores que se pueden utilizar sin ningún problema en la elección de la CA superior. En este caso, los valores de IN_i y OUT_i sólo le indicarían a la autoridad en qué momento actuar.

6.5 Ejemplo de Aplicación

Para mayor claridad, a continuación ejemplificamos el funcionamiento de PROSEARCH.

En la Figura 6.1 se muestran siete autoridades formando una PKI. Las flechas indican los certificados expedidos de una autoridad a otra. La autoridad 1 inicia el protocolo.

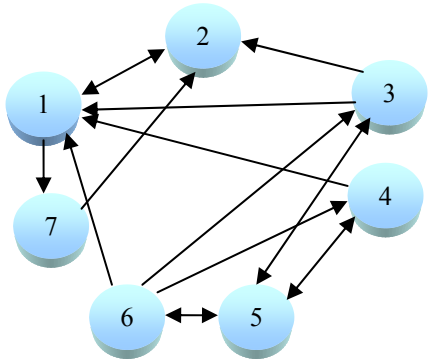

Figura 6.1: PKI ejemplo

6.5.1 Orden de Confiabilidad entre las CAs

Si la autoridad 1 desea colaborar para la creación de una jerarquía, en la primera ronda (*round*) envía un mensaje de solicitud a las CAs en que confía (2, 3, 4 y 6), así como a las

CAs que confían en ella (2 y 7). Además, anexa a esa solicitud su propuesta sobre la longitud máxima de los caminos de certificación, para lo cual vamos a suponer $L_{MAX} = 2$.

Una *ronda* es el conjunto de mensajes enviados y recibidos en el mismo instante de tiempo.

Supongamos que la autoridad 2 está dispuesta a colaborar con 1. Entonces, en la segunda ronda, envía un mensaje de aceptación a 1 y un mensaje de solicitud a sus demás CAs vecinas (3 y 7), junto con el valor de L_{MAX}.

Consideremos ahora el caso de la autoridad 7. Esta autoridad puede negarse a establecer la jerarquía porque no está de acuerdo con el valor de L_{MAX} o porque simplemente no está interesada en ser parte de la jerarquía. Por tanto, debe enviar, en la segunda ronda, un mensaje de rechazo a la autoridad 1.

Evaluemos ahora el caso de la autoridad 3. Ésta recibe un mensaje de solicitud de la autoridad 1. Como está dispuesta a colaborar, envía un mensaje de aceptación a 1 y una solicitud a 2, 5 y 6 en la segunda ronda.

Vamos a suponer que las demás autoridades quieren hacer parte de la jerarquía. Así, también en la segunda ronda, la autoridad 4 debe enviar un mensaje de aceptación a 1, y solicitar a 5 y 6 su colaboración. Por otra parte, la autoridad 6 envía un mensaje de aceptación a la autoridad 1 y un mensaje de solicitud a 3, 4 y 5.

En la tercera ronda, la autoridad 2 envía un mensaje de aceptación a la autoridad 3; la autoridad 3 envía un mensaje de aceptación a las autoridades 2 y 6; la autoridad 4 envía un mensaje de aceptación a la autoridad 6; la autoridad 5 envía un mensaje de aceptación a las autoridades 3, 4 y 6; la autoridad 6 envía un mensaje de aceptación a 3 y 4; y la autoridad 7 envía un mensaje de rechazo a la autoridad 2.

Después de recibir respuesta a todos los mensajes de solicitud enviados, las autoridades dispuestas a participar en la jerarquía deben determinar sus propios parámetros IN_i y OUT_i, y dárselos a conocer a sus vecinas participantes a través de un mensaje de información, lo que hacen durante la cuarta ronda. La Figura 6.2 indica los datos enviados por cada autoridad participante.

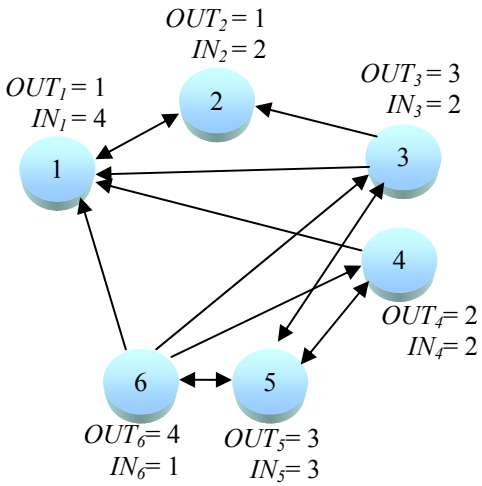

Figura 6.2: Información compartida

Cuando una autoridad conoce los parámetros de todas sus vecinas participantes, las ordena de la menos confiable a la más confiable y determina después de qué autoridades debe actuar en la segunda fase de PROSEARCH. Así, la autoridad 1 compara su parámetro OUT_1 con el de todas sus vecinas y las ordena de menor a mayor: 1, 2, 4, 3, 6. Pero OUT_1=OUT_2, por tanto, la autoridad 1 compara IN_1 e IN_2, determinando que la autoridad 2 es menos confiable. Por tanto, el orden de confiabilidad para la autoridad 1 será: 2, **1**, 4, 3, 6.

De igual manera, las demás autoridades establecen su orden de confiabilidad, que será para la autoridad 2: **2**, 1, 3; para la autoridad 3 será: 2, 1, **3**, 5, 6; para la autoridad 4 será: 1, **4**, 5, 6; para la autoridad 5 será: 4, 3, **5**, 6; y para la autoridad 6 es: 1, 4, 3, 5, **6**.

6.5.2 Creación de la Jerarquía

De acuerdo al orden establecido en la fase anterior, la primera autoridad en actuar será 2, luego actuará 1, después 3 y 4, posteriormente 5 y finalmente 6 que es la más confiable.

La autoridad 2 debe buscar entre las CAs en que confía (1 y 3), la que tiene el mayor OUT_i (es decir, la autoridad 3) y elegirla como su CA superior. La autoridad 2 le pregunta a la autoridad 3 el valor de L_3. L_3 es igual a L_2, por tanto, ahora $L_3 = L_2+1=1$, que es igual a (L_{MAX}–1)=1, lo que quiere decir que la autoridad 2 ha elegido una CA superior adecuada. Por tanto,

ahora el certificado expedido por la autoridad 3 a 2 cobra mayor importancia que el certificado expedido por la autoridad 1 a 2, como se ilustra en la Figura 6.3. Finalmente, en la quinta ronda, la autoridad 2 envía a sus vecinas participantes (1 y 3) un mensaje de asociación, indicando que ha elegido a 3 como CA superior y que el nuevo valor de L_3 es 1.

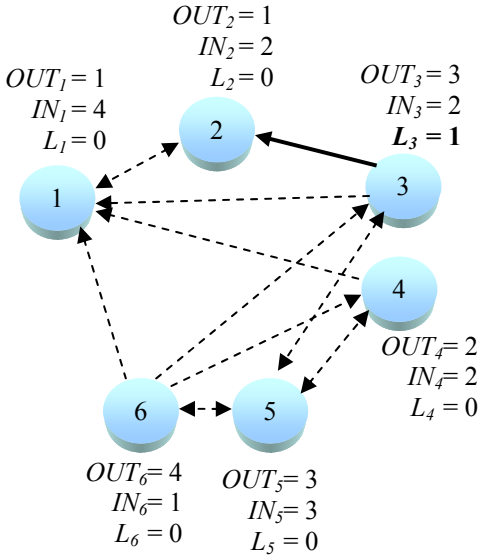

Figura 6.3: Autoridad 3 como CA superior de autoridad 2

La siguiente autoridad en actuar es 1. De las CAs en que confía (2, 3, 4 y 6), 6 es la que tiene el OUT_i más alto. Pero L_6 es igual a L_1, por tanto, L_6=L_1+1=1, que es igual a (L_{MAX}–1), así que la autoridad 6 es una CA superior adecuada para 1. Finalmente, en la sexta ronda, la autoridad 1 envía un mensaje de asociación a sus vecinas 2, 3, 4 y 6, con el valor de L_6.

El turno es ahora para las autoridades 3 y 4 que envían sus mensajes en la séptima ronda. La autoridad 3 confía en 6 y 5, pero OUT_6 es mayor que OUT_5, por tanto, 3 elige como CA superior a 6. L_6 es igual a L_3, así que L_6=L_3+1=2, que no cumple con la longitud máxima permitida (L_{MAX}–1)=1. Así, la autoridad 3 debe optar por elegir a la autoridad 5 como CA superior. L_5 es menor que L_3, por tanto, L_5=L_3+1=2, que es mayor que (L_{MAX}–1). Por consiguiente, la autoridad 3 debe enviar un mensaje de fallo a sus vecinas 1, 2, 5 y 6, y tendrá que repetir el protocolo con las otras raíces. Por otro lado, la autoridad 4 también confía en 5 y 6. Como L_4<L_6, L_6 continuará teniendo el mismo valor y 6 será la CA superior de 4.

Es el turno de la autoridad 5, que confía en 3, 4 y 6, de las cuales 6 es la que tiene el mayor OUT_i. L_5=0 que es menor que L_6, por tanto, 6 se convierte en la CA superior de 5 y la autoridad 5 se lo informa a sus vecinas en la octava ronda.

Por último, la autoridad 6 es la más confiable de su vecindario, así que envía un mensaje de CA raíz a sus vecinas (1, 3, 4 y 5) en la novena ronda. Como antes había recibido un mensaje de fallo de la autoridad 3, éste le indica que debe repetirse el protocolo entre las autoridades 3 y 6. La Figura 6.4 muestra la jerarquía y los parámetros resultantes antes de repetir el protocolo.

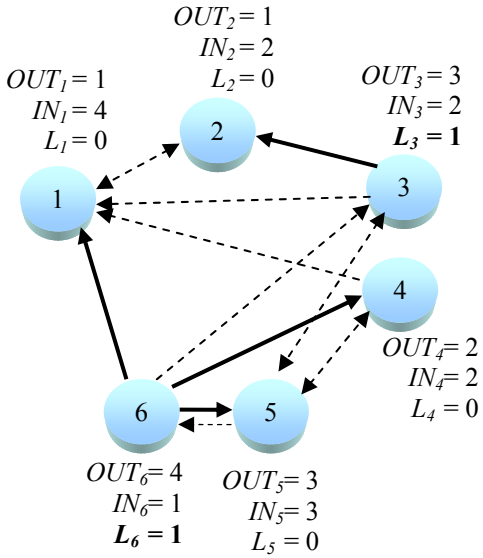

Figura 6.4: Jerarquía y parámetros resultantes

Cuando se repite el protocolo sólo interesan los certificados existentes entre 3 y 6, para determinar los nuevos valores de los parámetros IN_i y OUT_i. Por tanto, IN_3=1, OUT_3 =0, IN_6=0 y OUT_6=1. Esta información se envía durante la décima ronda. Así, el orden de confiabilidad para la autoridad 3 y la autoridad 6 será: 3, 6. Esto quiere decir que la autoridad 3 debe elegir a la autoridad 6 como CA superior. Como L_3 =L_6=1 entonces L_6=L_3+1=2, valor que es igual a L_{MAX}. Por tanto, la autoridad 6 se convierte en la CA superior de la autoridad 3 y ésta se lo informa a sus vecinas (1, 2, 5 y 6) en la undécima ronda.

La autoridad 6 se ha convertido en la raíz de la jerarquía, por lo que envía un mensaje de certificado raíz a sus CAs subordinadas (1, 3, 4 y 5) en la doceava ronda. Luego, la autoridad

3 retransmite este certificado a la autoridad 2, que es su CA subordinada, en la treceava ronda. La Figura 6.5 muestra la jerarquía establecida.

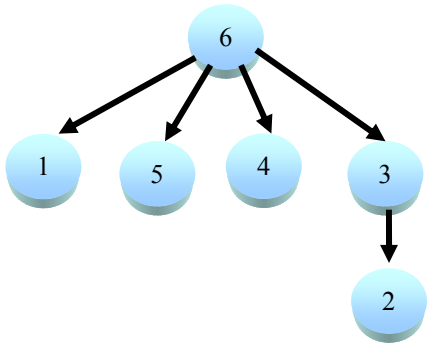

Figura 6.5: Jerarquía establecida

6.5.3 Tiempo de ejecución de PROSEARCH

El tiempo requerido para ejecutar PROSEARCH en el ejemplo anterior va a depender de la tecnología que empleen las CAs para comunicarse. Suponiendo que se trata de comunicaciones inalámbricas WLAN 802.11b (11Mbps), el tiempo requerido para una transmisión exitosa en este tipo de red, usando el esquema RTS/CTS, puede determinarse con los valores de la Tabla 6.2 y las ecuaciones (6.1), (6.2), (6.3), (6.4) y (6.5), tomadas de [CBV04]. Además, el caudal medio real (C) en este tipo de red, cuando existe un alto nivel de concurrencia es 2,739Mbps, de acuerdo a [ACG04]. Por tanto, si asumimos que la longitud máxima de un mensaje es 200 bytes (l=1600 bits) y el retardo de propagación es δ=1μs, la duración de una transmisión exitosa T_S^{RTS} es 1,34ms. Así que el tiempo necesario para llevar a cabo 13 rondas será 17,42ms, que es bastante corto.

$$T_S^{RTS} = DIFS+T_{RTS}+SIFS+T_{CTS}+SIFS+T_{header}+(l/C)+SIFS+T_{ACK}+4\delta \quad \textbf{(6.1)}$$

$$T_{header} = (MAC_{hdr}/C) + (PHY_{hdr}/C_{control}) \quad \textbf{(6.2)}$$

$$T_{ACK}=l_{ACK}/Cc_{ontrol} \quad \textbf{(6.3)}$$

$$T_{RTS}=l_{RTS}/Cc_{ontrol} \quad \textbf{(6.4)}$$

$$T_{CTS}=l_{CTS}/Cc_{ontrol} \quad \textbf{(6.5)}$$

Tabla 6.2: Parámetros del sistema DSSS usados en el estándar 802.11b

PARÁMETRO	VALOR
MAC header, MAC_{hdr}	272 bits
PHY header (long), PHY_{hdr}	192μs
RTS packet, l_{RTS}	160 bits + PHY_{hdr}
CTS packet, l_{CTS}	112 bits + PHY_{hdr}
ACK packet, l_{ACK}	112 bits + PHY_{hdr}
DIFS	50μs
SIFS	10μs
Control rate, $C_{control}$	2Mbit/s

También podemos calcular el tiempo requerido para firmar los mensajes enviados por cada autoridad y para verificar la firma de los mensajes recibidos. Si asumimos que cada CA es un ordenador con procesador Pentium 4 a 2,1GHz y Windows XP SP1, de acuerdo con [Dai04], el tiempo de ejecución de una operación de firma usando el algoritmo RSA-1024 es 4,75ms, mientras que el tiempo de ejecución de una operación de verificación es 0,18ms. Por tanto, el tiempo requerido para firmar y verificar la firma de 13 mensajes es 64,09ms.

Finalmente, el tiempo total requerido para ejecutar nuestro protocolo en el ejemplo citado es aproximadamente 81,51ms. Sin embargo, ya que este valor depende también del tiempo de respuesta de las entidades y la congestión de la red, debe ser tomado como un mínimo.

Por otra parte, el tiempo de ejecución de nuestro protocolo utilizando dispositivos limitados como PDAs y teléfonos móviles es más alto, así por ejemplo, si los terminales con los que operaran las entidades de la PKI fueran PDAs Compaq iPAQ H3630 con procesador StrongARM a 206MHz y Windows CE PC 2002, una operación de firma usando el algoritmo RSA-1024 duraría 78,25ms y una de verificación duraría 5,01ms, según se indica en [AVT04], por tanto, el tiempo necesario para ejecutar PROSEARCH en el ejemplo dado, es

aproximadamente 1,1 segundos, tiempo que aunque es mucho más grande que el anterior, todavía es razonable.

6.5.4 Búsqueda de Caminos Simplificada

Si la autoridad 2 de la Figura 6.2 quiere construir un camino de certificación a la autoridad 4, debe iniciar la búsqueda a partir de las CAs en que confía (1 y 3). En caso que busque el camino a través de la autoridad 1 sólo podrá encontrar la opción 1-2, ya que el único certificado que expide 1 es precisamente para la autoridad 2.

Por otra parte, si la autoridad 2 busca el camino a través de 3, encuentra las siguientes alternativas: 3-1-2, 3-2; 3-5-3, 3-5-4, 3-5-6-1-2, 3-5-6-3, 3-5-6-4, 3-5-6-5.

Así, de 9 posibles caminos sólo 2 conducen a la entidad objetivo (2/9).

Para validar el camino más corto (3-5-4), la autoridad 2 tendría que verificar la firma de dos certificados: el expedido por 3 a la autoridad 5 y el emitido por 5 a la autoridad 4. En cambio, si se emplea la jerarquía de la Figura 6.5, la autoridad 2 sólo tendrá que verificar la firma del certificado que 6 le expide a 4, con la clave pública de la autoridad 6, que es la base de confianza de la jerarquía. Además, gracias a las relaciones de confianza unidireccionales propias de la arquitectura jerárquica, existe sólo un único camino entre 2 y 4. La Tabla 6.3 muestra la relación entre el número de caminos exitosos y el número de caminos posibles de una autoridad a otra para la PKI de la Figura 6.2.

Tabla 6.3: Número de caminos exitosos vs. número de caminos posibles

DE \ A	1	2	3	4	5	6
1	X	7/41	7/19	6/22	5/13	5/32
2	5/11	X	1/2	2/9	1/4	1/7
3	6/18	6/18	X	4/12	3/6	3/14
4	10/22	10/22	4/12	X	3/7	3/17
5	7/12	7/12	2/8	2/10	X	1/6
6	13/24	13/24	3/11	4/15	3/7	X

6.5.5 Longitud de Caminos de Certificación

En esta sección comparamos la longitud de los caminos de certificación más cortos de la PKI representada en la Figura 6.2 y la jerarquía establecida con PROSEARCH en la Figura 6.5.

La longitud de un camino de certificación es el número de certificados que un verificador debe comprobar para autenticar otra entidad. La Tabla 6.4 muestra la longitud de los caminos más cortos entre cada par de entidades de la PKI considerada. Una longitud de camino igual a '0' significa que la CA conoce la clave pública de la otra autoridad, por lo que no es necesario verificar la firma de ningún certificado.

Por otra parte, la

Tabla 6.5 muestra la longitud de los caminos de certificación en la jerarquía establecida con nuestro protocolo. Los caminos que incrementan su longitud comparados con los valores en la Tabla 6.4, se encuentran resaltados. La longitud de los caminos restantes es menor o igual a la especificada en dicha tabla.

Tabla 6.4: Longitud de caminos más cortos en PKI ejemplo

DE \ A	1	2	3	4	5	6
1	0	0	0	0	1	0
2	0	0	0	2	1	2
3	1	1	0	1	0	0
4	1	2	1	0	0	0
5	1	1	0	0	0	0
6	1	2	1	1	0	0

Tabla 6.5: Longitud de caminos con PROSEARCH

DE \ A	1	2	3	4	5	6
1	0	2	1	1	1	0
2	1	0	0	1	1	0
3	1	1	0	1	1	0
4	1	2	1	0	1	0
5	1	2	1	1	0	0
6	1	2	1	1	1	0

6.5.6 Vinculación de una Nueva CA a la Jerarquía

La autoridad 8 quiere vincularse a la jerarquía (ver Figura 6.6), por tanto pregunta a 1, 3 ó 5 el valor de L_{MAX}. Como está de acuerdo con este valor puede unirse a la jerarquía, y se lo comunica a sus vecinas participantes (1, 3 y 5) a través de un mensaje de solicitud. Estas autoridades deben calcular de nuevo sus parámetros IN_i y OUT_i y enviárselos a sus vecinas en un mensaje de información, al igual que la autoridad 8. Las vecinas les enviarán también sus correspondientes parámetros, prosiguiendo con la primera fase del protocolo. La Figura 6.7 muestra los nuevos parámetros de todas las autoridades.

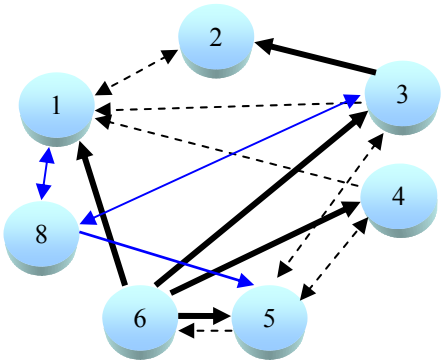

Figura 6.6: Vinculación a la jerarquía

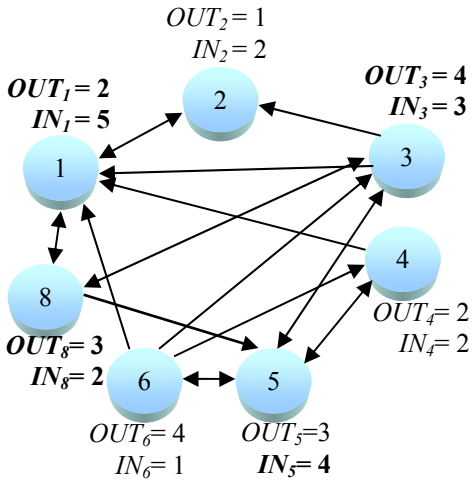

Figura 6.7: Parámetros de las CAs

Por tanto, el orden de confiabilidad para la autoridad 1 es: 2,4, **1**, 8, 6, 3; para la autoridad 2 es: **2**, 1, 3; para la autoridad 3 es; 2, 1, 8, 5, 6, **3**; para la autoridad 4 es; **4**, 1, 5, 6; para la autoridad 5 es: 4, 8, **5**, 6, 3; para la autoridad 6 es: 4, 1, 5, **6**, 3; y para la autoridad 8 es: 1, **8** ,5, 3.

Así, las autoridades 2 y 4 actúan primero en la segunda fase del protocolo, luego la autoridad 1, más tarde la autoridad 8, después la autoridad 5, enseguida de la autoridad 6 y finalmente la autoridad 3 es la más confiable.

La autoridad 2 vuelve a elegir a 3 como CA superior, pues es su vecina más confiable y L_3=1. La autoridad 4, por su parte, elige otra vez a la autoridad 6 como CA superior y L_6=1.

Luego, la autoridad 1 elige a 3 como CA superior por ser ahora su vecina más confiable. Como $L_1<L_3$=1, que es igual a (L_{MAX}–1), ésta es una CA superior apropiada.

La autoridad más confiable para 8 es 3, y como $L_8<L_3=(L_{MAX}–1)$, 8 puede elegir a la autoridad 3 como CA superior.

La siguiente autoridad en actuar es 5, cuya vecina más confiable ahora es 3, así que la elige como CA superior y L_3 sigue igual.

Ahora actúa la autoridad 6, que sólo confía en la autoridad 5, cuyo orden de confiabilidad es menor. Por tanto, el protocolo debe repetirse entre las autoridades 3 y 6. Cuando esto sucede, la autoridad 6 resulta ser más confiable que la autoridad 3, así que 3 debe elegirla como CA superior. La autoridad 6 será entonces la CA raíz de toda la jerarquía (Figura 6.8) y L_6 será igual a 2.

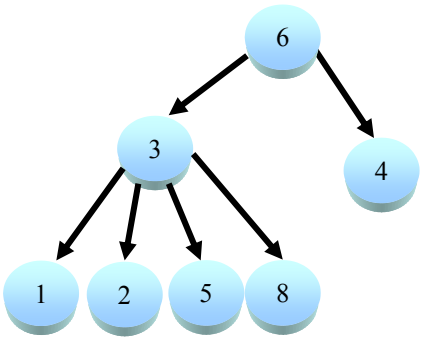

Figura 6.8: Nueva jerarquía incorporando la autoridad 8

6.5.7 Salida de una CA de la Jerarquía

Supongamos que la autoridad 3 falla o sale de la jerarquía por algún motivo. En este caso, tan pronto como sus CAs subordinadas se den cuenta de que la autoridad 3 no está disponible deben elegir una nueva CA superior y comunicárselo a sus vecinas, especialmente a las autoridades que están por debajo de ellas en la jerarquía, para que estas también tengan la opción de cambiar de CA superior si así lo desean. En nuestro caso particular, las autoridades 1, 2, 5 y 8 deberán elegir una nueva CA superior. Para ello estas autoridades y sus vecinas inician de nuevo el protocolo determinando el valor de sus parámetros IN_i y OUT_i (Figura 6.9)

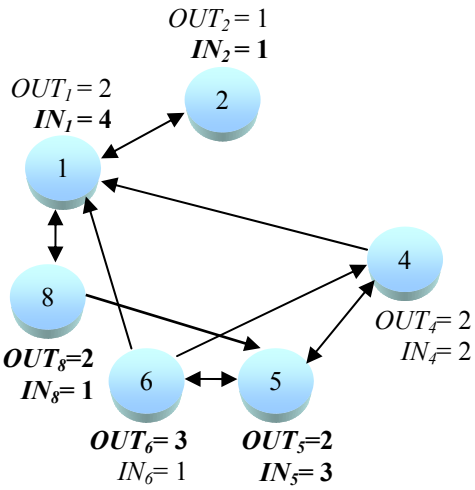

Figura 6.9: PKI sin autoridad 3

El nuevo orden de confiabilidad para la autoridad 1 es: 2, 8, 4, **1**, 6; para la autoridad 2 es: **2**, 1; para la autoridad 4 es: **4**, 5, 1, 6; para la autoridad 5 es: 8, 4, **5**, 6; para la autoridad 6 es: 4, 5, 1. **6**; y para la autoridad 8 es: **8**, 5, 1. Por tanto, las autoridades 2, 4 y 8 actúan primero en la segunda fase del protocolo, luego 1 y 5, y finalmente la autoridad 6 es la más confiable.

Las autoridades 2 y 8 eligen a 1 como CA superior y L_1 va a ser igual a 1. Por su parte, la autoridad 4 no va a cambiar de CA superior puesto que 6 sigue siendo su vecina más confiable, por tanto, L_6=1.

La autoridad 1 no puede elegir a 6 como CA superior puesto que L_6 sería mayor que (L_{MAX}–1). La autoridad 5, en cambio, elige a 6 como CA superior y L_6 no modifica su valor.

Finalmente, las autoridades 1 y 6 repiten el protocolo. La autoridad 1 elige a 6 como CA superior y ésta se convierte en la raíz de la jerarquía como se ilustra en la Figura 6.10.

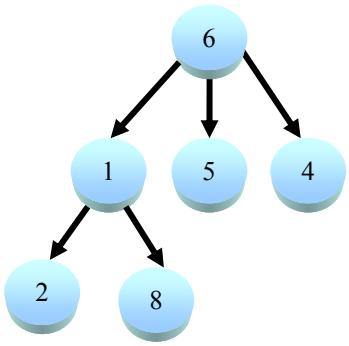

Figura 6.10: Nueva jerarquía sin la autoridad 3

6.5.8 Parámetros Falsos

Si la autoridad 4 de la Figura 6.9 es una entidad maliciosa y decide enviar parámetros falsos a sus vecinas durante la primera fase del protocolo, por ejemplo, IN_4=4 y OUT_4=4, para intentar ser la raíz de la jerarquía, en la segunda fase del protocolo, las primeras autoridades en actuar serán 2 y 8, luego actuarán 1 y 5, más tarde 6 y finalmente 4.

Las autoridades 2 y 8 elegirán a 1 como CA superior por ser su vecina más confiable y L_1 será igual a 1. Luego, la autoridad 1 no podrá elegir a 4 o a 6 como CA superior porque sus L_i serían mayores que (L_{MAX}–1), por tanto, debe repetir el protocolo. La autoridad 5, por su parte, elige a 4 como CA superior y L_4 tomaría el valor de 1. Finalmente, la autoridad 6 no puede elegir una CA superior porque 5 que es el único vecino en que confía es menos confiable que ella. Por tanto, las autoridades 1, 4 y 6 deben repetir el protocolo.

Cuando se repite el protocolo, la autoridad 4 no puede volver a enviar parámetros falsos porque 1 y 6 podrían detectarlo gracias a la relación de confianza que existe entre ellas y a que todas pertenecen al mismo vecindario. Así, los parámetros de las autoridades serán: IN_1=2, OUT_1=0, IN_4=1, OUT_4=1, IN_6=0 OUT_6=2. Las autoridades 1 y 4 elegirán entonces a 6 como CA superior por ser la más confiable. La autoridad 6 se convertirá en la raíz de la jerarquía y la autoridad 4 no habrá podido cumplir con su objetivo. La Figura 6.11 muestra la jerarquía establecida.

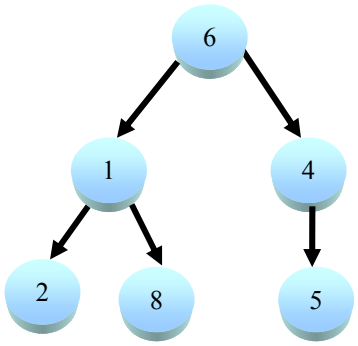

Figura 6.11: Jerarquía con autoridad maliciosa

6.6 PROSEARCH en Arquitecturas de Autorización

Las arquitecturas de autorización deben emplear algún tipo de servicio de autenticación, para funcionar adecuadamente y PKI puede proporcionarles este servicio. Cuando el modelo de confianza utilizado por estas arquitecturas es en malla, la búsqueda de caminos de certificación por parte del verificador va a ser más rápida si se crea una jerarquía virtual entre las CAs de la PKI utilizando PROSEARCH. Esto contribuirá a la eficiencia del servicio de autorización.

6.7 Conclusiones

Las relaciones de confianza bidireccionales incrementan la complejidad del proceso de construcción de caminos, puesto que todas las opciones no llevan a la entidad objetivo y pueden existir múltiples caminos entre dos entidades.

En este capítulo proponemos un protocolo al que hemos llamado PROSEARCH. Este protocolo establece una jerarquía virtual entre las autoridades de una PKI con relaciones de confianza bidireccionales, basado en la confiabilidad de las CAs participantes. El nivel de confiabilidad de cada CA es establecido de acuerdo a dos parámetros: el número de autoridades en que confía o le emiten un certificado (IN_i) y el número de entidades que confían en ella o a las que les emite un certificado (OUT_i).

Una ventaja de PROSEARCH es que no establece nuevas relaciones de confianza entre las CAs sino que establece la jerarquía utilizando las relaciones existentes. Por tanto, no es necesario expedir nuevos certificados entre las autoridades.

El establecimiento de una jerarquía entre las autoridades facilita el proceso de construcción de caminos, tal como se ejemplifica en la sección 6.5.3, ya que las relaciones de confianza son unidireccionales y existe un único camino entre dos autoridades. Esto contribuye a incrementar la eficiencia del proceso de validación de caminos.

Nuestro protocolo es adaptable a usuarios con limitada capacidad de procesamiento y almacenamiento, ya que la jerarquía se construye considerando una longitud máxima de los caminos (L_{MAX}). Es la autoridad que inicia el protocolo quien establece el valor de este parámetro y lo puede fijar de acuerdo a las características de los terminales de sus usuarios.

La

Tabla 6.5 muestra que la jerarquía establecida con nuestro protocolo establece en la mayoría de los casos el camino más corto entre dos autoridades.

PROSEARCH garantiza también la autenticidad de los mensajes que comparten las autoridades, gracias a que estos son firmados digitalmente por la CA que los emite.

Entre los posibles ataques a nuestro protocolo está el que las autoridades envíen datos falsos a sus vecinas, aunque esto teóricamente no debería ocurrir ya que estamos tratando con terceras partes de confianza.

Nuestro protocolo no siempre encuentra una única raíz para la jerarquía, lo que no implica que no exista camino entre las raíces resultantes. Por esta razón, cuando esto suceda, recomendamos usar métodos alternativos de búsqueda de caminos para encontrar el camino más corto entre estas entidades.

Capítulo 7

Evaluación de PROSEARCH en Diferentes Entornos

7.1 Introducción 143
7.2 PROSEARCH en Infraestructuras Críticas 144
7.3 PROSEARCH en Redes Ad-Hoc 146
7.4 Conclusiones 162

7.1 Introducción

El capítulo 6 describió el funcionamiento de PROSEARCH y sus principales ventajas, como son: simplificar la búsqueda de caminos, no requerir la expedición de nuevos certificados y rápida ejecución. Estas características hacen que nuestro protocolo sea adaptable a diferentes escenarios, como se evidencia en [SPF06b] y [SHF], donde proponemos el uso de PROSEARCH en infraestructuras críticas y redes ad-hoc, respectivamente. Evaluamos además el desempeño de nuestro protocolo en los dos entornos. El contenido de estos artículos se expresa a lo largo de este capítulo.

En la sección 7.2 vemos como se comporta PROSEARCH cuando se utiliza en infraestructuras críticas. La sección 7.3 describe las diferentes propuestas que han contribuido a acercar la tecnología PKI a las redes ad-hoc y evalúa el desempeño de PROSEARCH en este tipo de redes. Finalmente la sección 7.4 concluye.

7.2 PROSEARCH en Infraestructuras Críticas

A medida que las redes de comunicaciones han incrementado su importancia en la vida cotidiana, nuestra dependencia de su infraestructura ha crecido también. Desafortunadamente, al mismo tiempo, los ataques hostiles a esta infraestructura han incrementado en número e impacto. Así pues, las redes son cada vez más vulnerables y es esencial garantizar la seguridad de la información que se considera crítica desde el punto de vista político, económico, financiero, social, etc.

En algunos escenarios críticos, como en los sistemas de recuperación de desastres o las aplicaciones militares, se requiere el despliegue rápido de una red segura, en la que exista un camino entre cada par de nodos y todos puedan autenticarse mutuamente. Las PKIs en malla pueden proporcionar este servicio de autenticación gracias a su dinamismo y a que los caminos de certificación pueden construirse aunque parte de la infraestructura esté temporalmente inalcanzable, lo que es común después de desastres o ataques de red. Sin embargo, el descubrimiento de los caminos de certificación es difícil en estas PKIs, y PROSEARCH puede ayudar a simplificar este proceso. Además, su corto tiempo de ejecución hace que se reconfigure la jerarquía rápidamente tras un desastre.

7.2.1 Evaluación

En esta sección evaluamos el desempeño de PROSEARCH en una infraestructura crítica. La Figura 7.1 compara el número de rondas necesitadas para llevar a cabo PROSEARCH con todas las CAs operando correctamente (sin fallos) y después de un desastre, cuando el 30% de las CAs dejan de operar (con fallos). Por otra parte, la Figura 7.2 muestra el número de raíces resultantes en cada caso. Las tablas de datos de estas figuras se encuentran en el apéndice A, Tablas A.12 y A.13.

Los resultados mostrados en las Figuras 7.1 y 7.2 fueron obtenidos calculando el promedio de 50 iteraciones por cada combinación de los siguientes parámetros:

- Número de CAs: De 10 a 200, en incrementos de 10.
- Proporción de relación: 0,2; 0,4 y 0,6
- Longitud máxima permitida (L_{MAX}): 3
- Porcentaje de fallos en la red: 30%

La proporción de relación (rel) es la probabilidad de que una autoridad confíe en otra, es decir, la posibilidad de que una CA reciba un certificado de otra CA en la red.

El porcentaje de fallos en la red significa que el 30% de las CAs fallan después de utilizar PROSEARCH, así que es necesario reconfigurar la jerarquía con las CAs que siguen operando. Las CAs que fallan se eligen aleatoriamente.

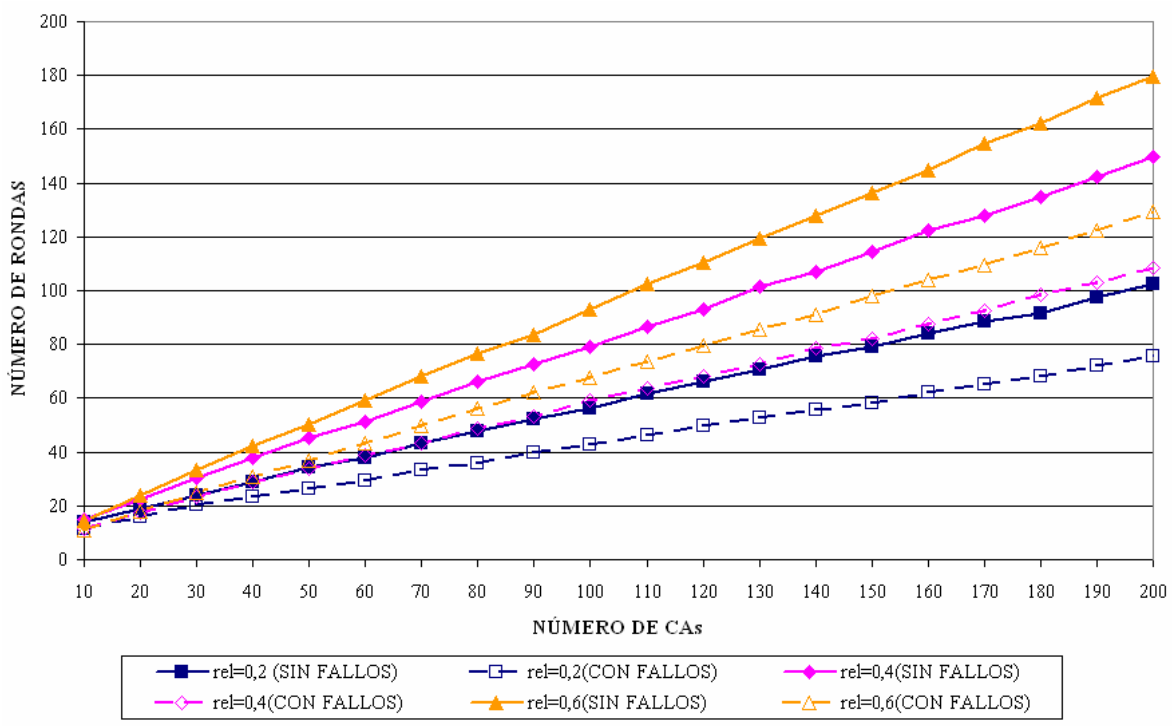

Figura 7.1: Número de CAs vs. número de rondas

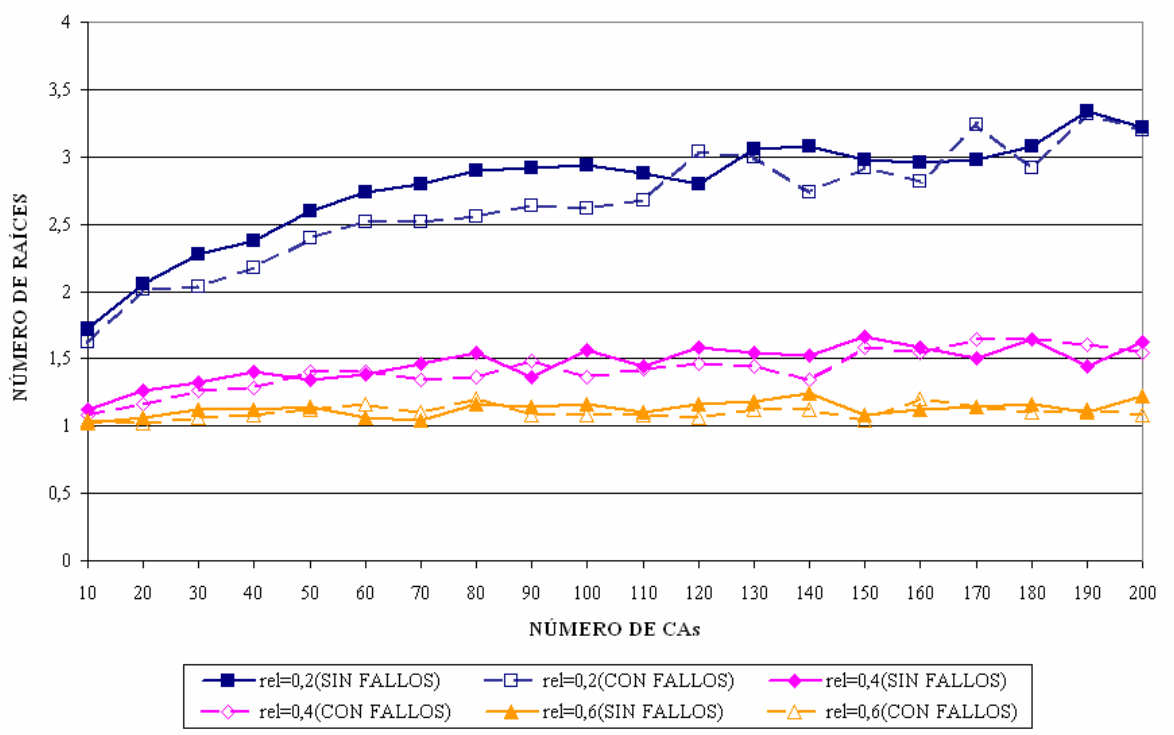

Figura 7.2: Número de CAs vs. número de CAs raíz

7.3 PROSEARCH en Redes Ad-Hoc

El desarrollo de las redes inalámbricas les ha permitido a los usuarios móviles, compatibles entre sí, establecer redes de corta duración para satisfacer sus necesidades de comunicación en determinado momento llamadas redes ad-hoc. Las MANETs (Mobile Ad-hoc Networks) son redes peer-to-peer, sin ningún tipo de infraestructura preestablecida, donde los nodos pueden moverse libremente y organizarse arbitrariamente. Además, son redes altamente dinámicas, ya que los nodos entran y salen frecuentemente de ellas. Por tanto, el establecimiento de relaciones de confianza entre los nodos de una MANET requiere protocolos rápidos y flexibles, independientes de una infraestructura de red fija.

PKI ha sido diseñada primordialmente para redes centralizadas, cableadas y bien conectadas, de manera que adoptar PKI en MANETs no es una tarea fácil. Además, en el entorno móvil, es muy probable el compromiso de algún nodo, por lo que se requieren ciertos pasos preventivos para garantizar la seguridad de los certificados de clave pública.

Adicionalmente, cuando determinadas CAs no estén disponibles, algunos servicios de la PKI, como el procesamiento de caminos de certificación y el chequeo de estado de los certificados, tampoco lo estarán.

Estos inconvenientes han sido resueltos distribuyendo la funcionalidad de la CA [YK03] [DRW04] y utilizando PKIs auto-organizadas [HBC01]. En este último enfoque, se debe tener en cuenta la gestión de múltiples relaciones de confianza, ya que dichas relaciones pueden ser bidireccionales. Esto dificulta el descubrimiento de los caminos de certificación.

PROSEARCH simplifica la construcción de los caminos de certificación, estableciendo una jerarquía virtual. La rapidez con que PROSEARCH establece esta jerarquía hace que se adapte al entorno altamente dinámico de las MANETs. Además, no requiere la expedición de nuevos certificados entre los participantes en la jerarquía virtual y la longitud máxima de los caminos puede adaptarse a la capacidad limitada que tienen algunas veces los terminales de los usuarios.

7.3.1 PKI y Redes Ad-hoc

Debido al rango limitado y poca confiabilidad de las conexiones inalámbricas, las redes ad-hoc móviles no pueden garantizar la conectividad entre sus nodos. La ausencia de una infraestructura de ruteo que asegure la conectividad entre nodos, impide el soporte de una infraestructura de confianza estable y duradera. Además, cuando un solo nodo móvil funciona como CA, éste puede actuar como punto único de fallo en caso de verse comprometido, o puede dejar fuera de servicio a toda la red si sale de ella. La duplicación de las autoridades puede usarse para evitar este problema, pero no es una solución escalable desde el punto de vista administrativo y el compromiso de cualquiera de ellas introduce diversos puntos de fallo.

Existen esencialmente dos enfoques que eliminan el uso de una sola autoridad de certificación en las redes ad-hoc móviles. El primero consiste en distribuir la funcionalidad de la autoridad de certificación entre varios nodos usando criptografía umbral; y el segundo se basa en una infraestructura de clave pública auto-organizada.

7.3.1.1 CA Distribuida con Criptografía Umbral

La facilidad con que los nodos pueden entrar y salir de una MANET muestra que es aconsejable distribuir la funcionalidad de la CA entre varios nodos de la red en lugar de asignársela a un solo nodo. Usando criptografía umbral se requiere un determinado número de nodos confiables para firmar un certificado, por lo que un adversario necesitaría corromper muchos de ellos para falsificar dicho certificado.

La criptografía umbral consiste fundamentalmente en compartir de manera segura un secreto. Este esquema de distribución de secretos permite dividir un secreto entre varios nodos de manera que se cumplan los siguientes requerimientos: (1) ningún grupo de nodos corruptos (menor que un umbral dado) puede reconstruir el secreto, incluso si ellos colaboran entre sí; (2) cuando sea necesario reconstruir el secreto, un gran número de nodos (mayor que el umbral ya nombrado) puede siempre hacerlo. Muchos sistemas de criptografía umbral existentes se basan en la técnica de distribución de secretos (k,n) de Shamir [Sha79]. En estos sistemas, la criptografía umbral se usa para distribuir la clave privada de la CA entre n nodos, pero la funcionalidad de la CA puede ser asumida sólo por k nodos ($k<n$). Así, k nodos pueden colaborar para generar una firma digital.

Las propuestas que permiten distribuir la funcionalidad de la CA en MANETs pueden clasificarse en dos tipos: CAs parcialmente distribuidas [YK03] [BG04] [DGS05] y CAs completamente distribuidas [KZL01] [DRW04] . En el primer caso, la potencia de firmado es distribuida entre un conjunto de nodos; y en el segundo, se distribuye entre todos los miembros del grupo.

7.3.1.2 PKI Auto-Organizada

En PGP (Pretty Good Privacy), cualquier usuario puede certificar, a través de su firma digital, la clave pública de otro usuario. Estos certificados forman una red de relaciones de confianza igual a igual, llamada con frecuencia *web of trust* [Zim05]. Sin embargo, la distribución de estos certificados se basa en directorios públicos que residen en servidores

gestionados de forma centralizada. Por tanto, este enfoque no es completamente auto-organizado.

Hubaux et al. [HBC01] proponen una implementación distribuida de PGP donde los certificados sean guardados y distribuidos por los propios usuarios. Así, cada usuario cuenta con un repositorio de certificados local que contiene un número limitado de certificados. Cuando un usuario desea obtener la clave pública de otro, estos combinan sus repositorios de certificados locales para encontrar un camino de certificación apropiado. Los autores presentan dos algoritmos que los usuarios pueden usar para construir sus repositorios de certificados locales y muestran como estos algoritmos tienen un alto desempeño en grafos de confianza PGP reales, lo que significa que existe una alta probabilidad de que cualquier par de usuarios encuentre cadenas de certificados de uno al otro usando sólo sus repositorios de certificados locales. Además, el tamaño de los repositorios de certificados locales es pequeño comparado con el número total de usuarios en el sistema.

PROSEARCH asume que existe una PKI auto-organizada entre los nodos de la MANET como proponen Hubaux et al.

7.3.2 Evaluación

Para evaluar el funcionamiento de PROSEARCH en un ambiente con muchos dispositivos, hemos desarrollado un entorno de prueba que se basa en las siguientes suposiciones:

- Los parámetros IN_i y OUT_i de los nodos son generados aleatoriamente con una proporción de relación (rel) predefinida.
- El protocolo de enrutamiento de la red ad-hoc siempre encuentra el camino más corto (en saltos de red) entre dos nodos. PROSEARCH es independiente del protocolo de enrutamiento y sólo necesita conocer el número de saltos de red entre los nodos.
- El tiempo necesario para establecer una jerarquía es medido en rondas. El tiempo de duración de una ronda será diferente cada vez, pues depende del tiempo de procesamiento de los nodos, la velocidad de la red, la latencia, etc.

- Las entidades se sitúan aleatoriamente en una superficie de 20x20 m^2.
- El rango de cobertura de cada entidad es 30m.

Los resultados mostrados a continuación han sido obtenidos calculando el promedio de 50 iteraciones por cada combinación de los siguientes parámetros:

- Número de entidades: de 10 a 200, en incrementos de10.
- Proporción de relación: 0,1; 0,3; 0,4; 0,5, 0,7; 0,9
- Longitud máxima de los caminos permitida (L_{MAX}): de 2 a 10, en incrementos de 1.

7.3.2.1 Longitud Máxima de los Caminos de Certificación

La Figura 7.3 muestra la longitud máxima alcanzada por los caminos de certificación a medida que incrementa la proporción de relación. Para ello, determinamos la mejor jerarquía que se podía obtener en cada caso, a partir de las relaciones de confianza existentes, y establecimos la longitud del camino más largo. La tabla de datos de esta figura se encuentra en el apéndice A, Tabla A.14

Según podemos ver en la Figura 7.3, la longitud máxima de los caminos es menor que 4, por lo que tomaremos este valor como L_{MAX} de referencia de aquí en adelante.

7.3.2.2 Variación de la Proporción de Relación

Vamos a mantener la longitud máxima de los caminos constante, L_{MAX}=4, y a incrementar de 0,1 a 0,9 la proporción de relación. La Figura 7.4 muestra que nuestro protocolo tiende a encontrar una CA raíz cuando la proporción de relación es mayor que 0,4. Además, si la proporción de relación es 0,3, el número promedio de CAs raíz es 2. También puede verse que el número de raíces cuando la proporción de relación es mayor o igual que 0,3 es independiente del número de CAs, pues su valor cambia muy levemente a medida que el número de CAs aumenta. La tabla de datos de esta figura se encuentra en el apéndice A, Tabla A.15.

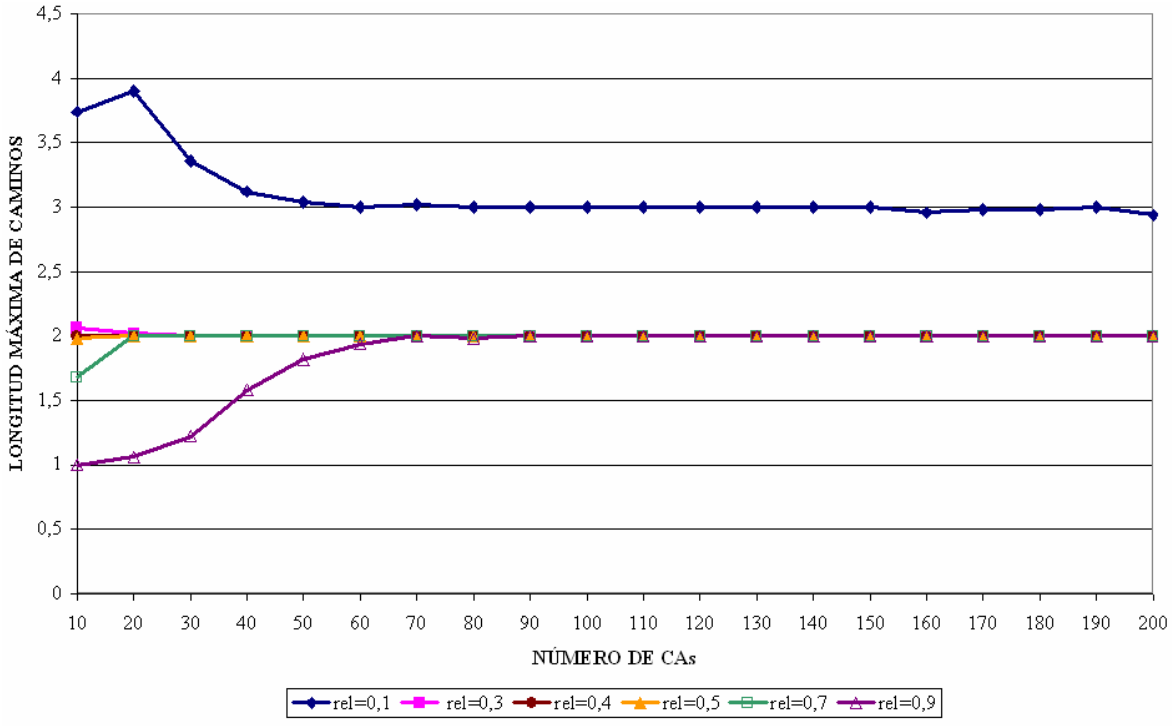

Figura 7.3: Longitud máxima de los caminos con mejor jerarquía

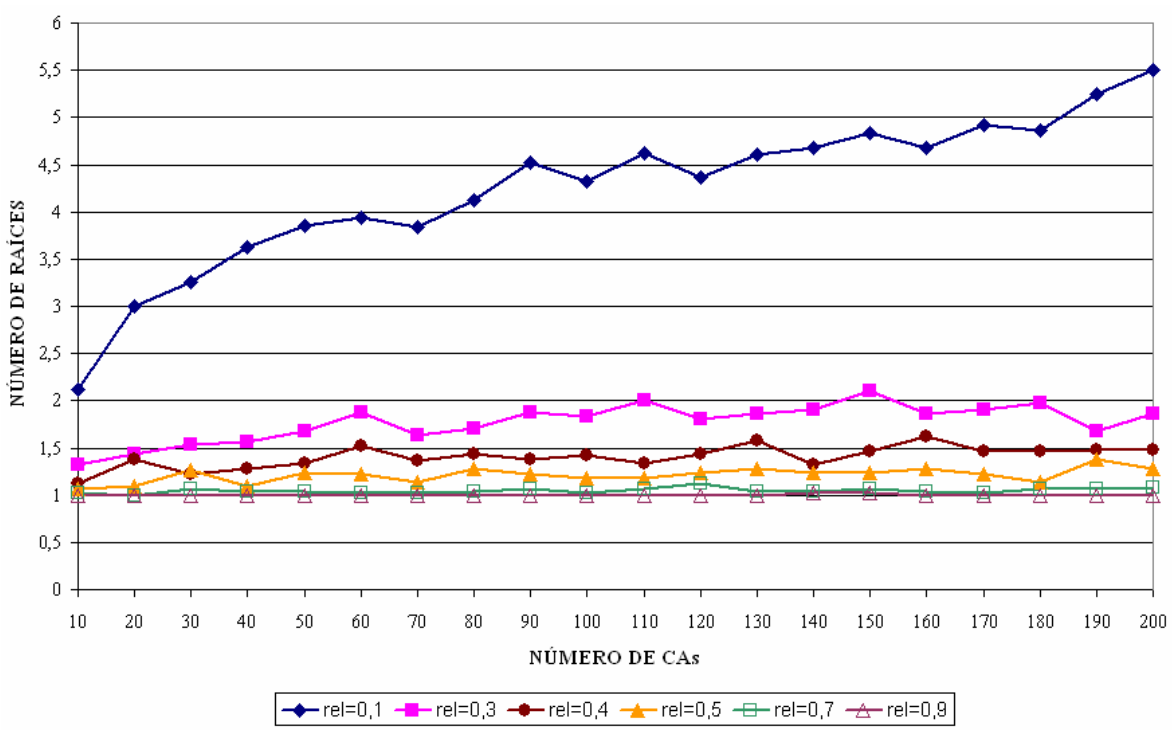

Figura 7.4: Número de CAs vs. número promedio de raíces

La Figura 7.5 muestra que la longitud de los caminos de certificación no sobrepasa la longitud máxima permitida (L_{MAX}=4). A medida que la proporción de relación aumenta de 0,1 a 0,9, la longitud promedio de los caminos decrece. Si la proporción de relación fuera 1, la longitud promedio de los caminos alcanzaría su valor mínimo porque existe una CA raíz y las otras autoridades se asocian directamente a esta raíz. Así, el número de certificados de la raíz a las hojas es 1. Sin embargo, si la proporción de relación es igual a 1, no tiene sentido llevar a cabo nuestro protocolo ya que el camino de una entidad a otra es fácil de encontrar.

La Figura 7.5 puede compararse con la Figura 7.6 que muestra la longitud promedio de los caminos obtenida con la mejor jerarquía posible en cada caso. Como se puede observar, la longitud promedio de los caminos con nuestro protocolo es levemente mayor.

Las tablas de datos de estas figuras se encuentran en el apéndice A, Tablas A.16 y A.17.

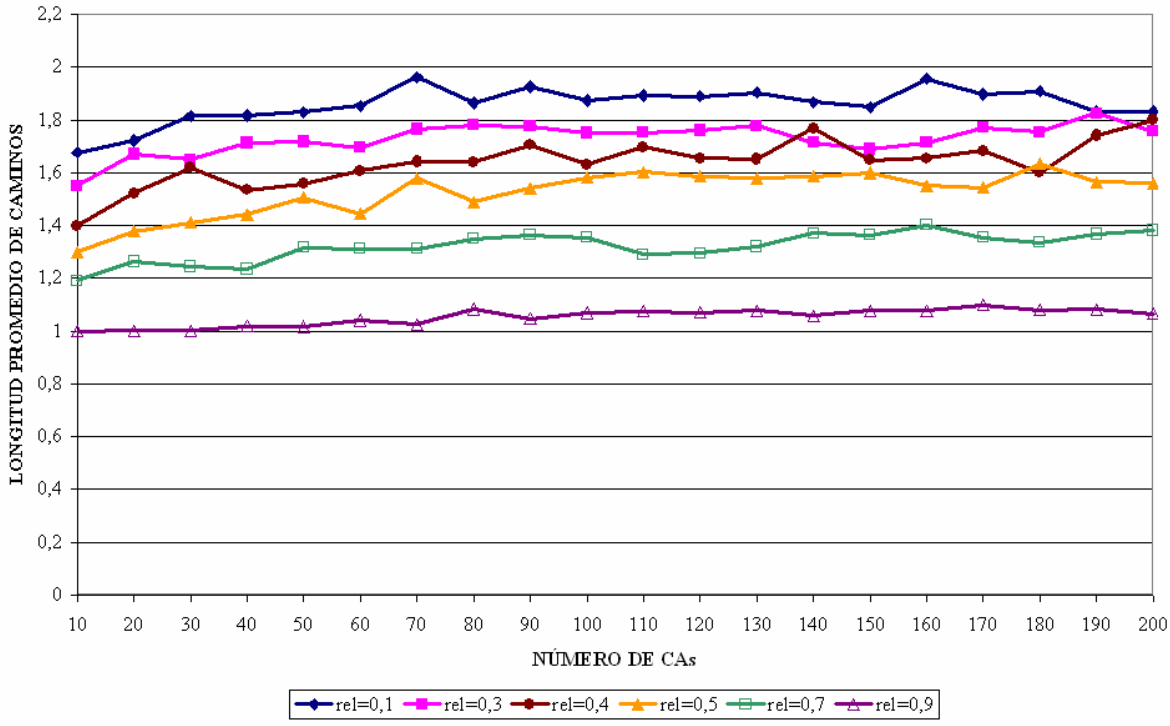

Figura 7.5: Número de CAs vs. longitud promedio de los caminos

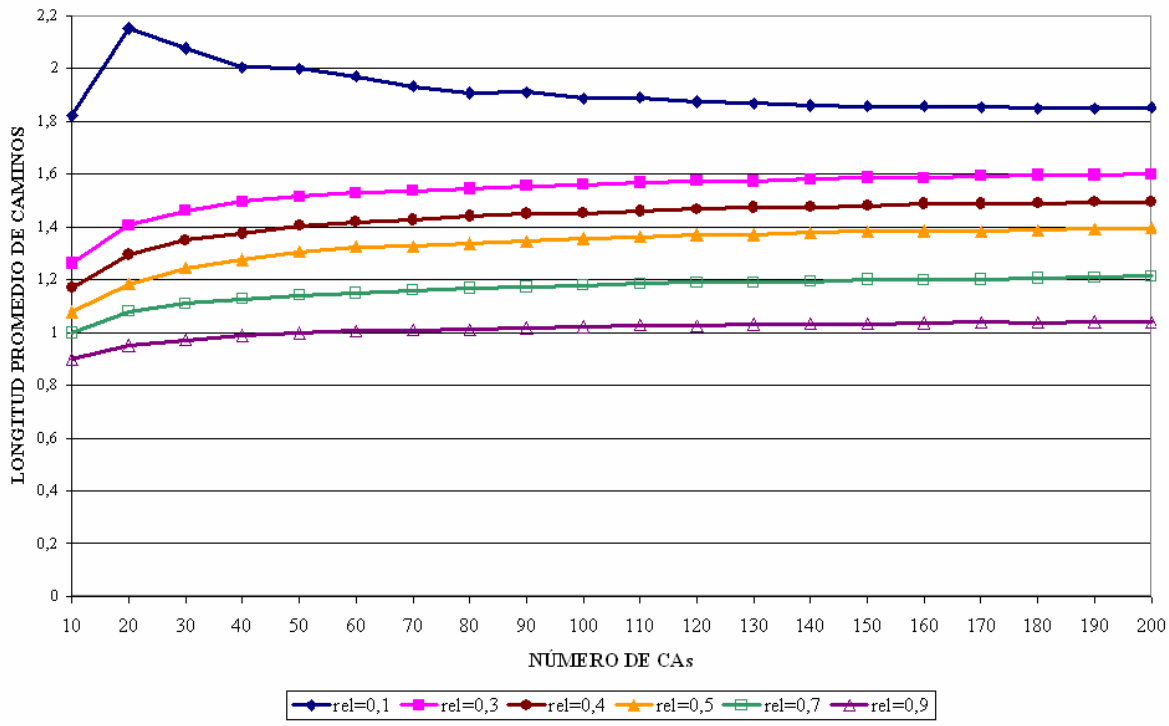

Figura 7.6: Longitud promedio de caminos con mejor jerarquía

También, evaluamos la probabilidad de fallo (P_e) de la jerarquía establecida, cuando alguna entidad quiera encontrar un camino a cualquier otra entidad. Cuando existe una sola CA raíz en la jerarquía, la probabilidad de fallo de nuestro protocolo es cero, ya que existe un camino entre cada par de entidades. Por otra parte, cuando existen varias CAs raíz, la probabilidad de fallo dependerá del número de CAs en cada subárbol, por lo que la probabilidad de fallo podría definirse como la sumatoria de la probabilidad de pertenecer a un subárbol por la probabilidad de no pertenecer al mismo (ver ecuación 7.1).

$$P_e = \sum_{s=1}^{r} P_s (1 - P_s) \tag{7.1}$$

Donde:

P_e: Probabilidad de fallo.

r: Número de CAs raíz.

s: identificador de subárbol.

P_s: Probabilidad de pertenecer al subárbol s.

Por ejemplo, si existen 3 CAs raíz en una red de 10 entidades y la primera CA raíz tiene 4 entidades subordinadas (subárbol 1), la segunda tiene 2 entidades subordinadas (subárbol 2) y la tercera tiene 1 entidad subordinada (subárbol 3), la probabilidad de fallo de nuestro protocolo será la probabilidad de que una entidad en el subárbol 1 desee encontrar un camino a alguna de las entidades del subárbol 2 y del subárbol 3, más la probabilidad de que una entidad del subárbol 2 quiera encontrar un camino a alguna entidad en los subárboles 1 y 3, más la probabilidad de que una entidad en el subárbol 3 quiera encontrar un camino a alguna entidad en los subárboles 1 y 2, como se muestra a continuación

$$P_e=5/10(3/10+2/10) + 3/10(5/10+2/10)+2/10(5/10+3/10)=0,62$$

La Figura 7.7 muestra que la probabilidad de fallo es menor del 10% cuando la proporción de relación es mayor que 0,5, independientemente del número de CAs. Cuando la proporción de relación es 0,4, la probabilidad de fallo es menor del 15%, lo que significa que el 85% de las CAs descubren fácilmente un camino entre ellas usando la jerarquía establecida con nuestro protocolo. Esta probabilidad de fallo se duplica (30%) cuando la proporción de relación es 0,3 y alcanza valores mucho más altos cuando la proporción de relación es igual a 0,1, debido a que existen diversas CAs raíz que no pertenecen al mismo vecindario y que por tanto desconocen de la existencia de las otras. La tabla de datos de esta figura se encuentra en el apéndice A, Tabla A.18.

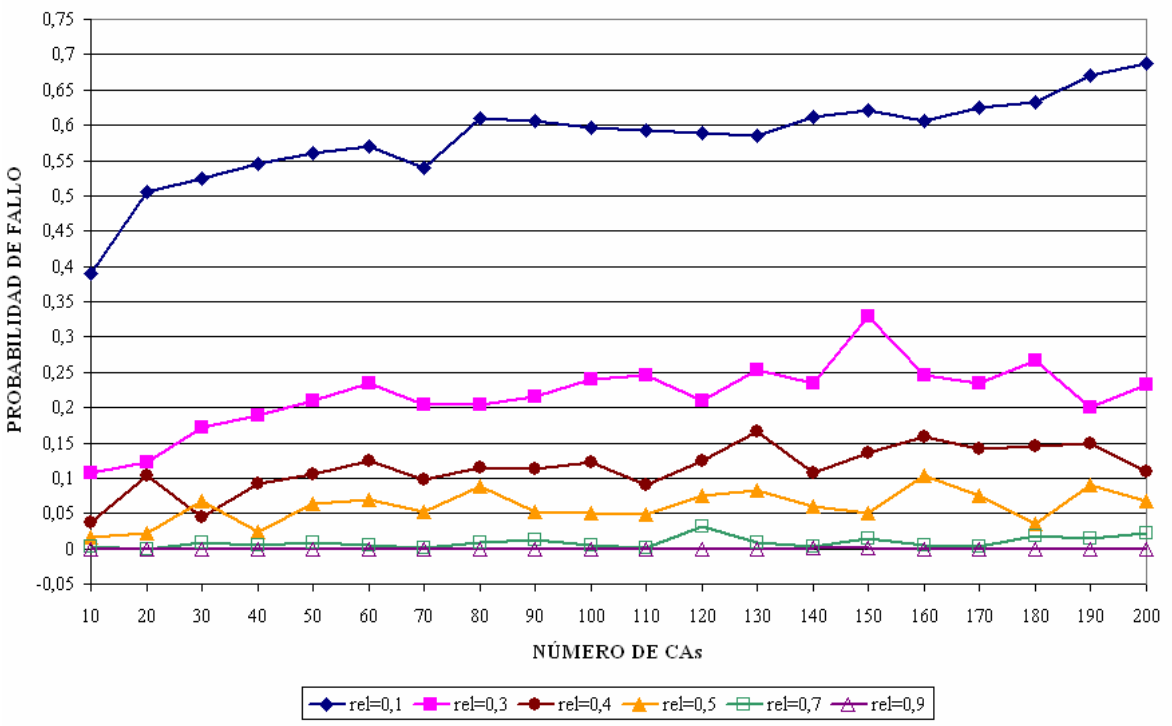

Figura 7.7: Número de CAs vs. probabilidad de fallo

La Figura 7.8 muestra que el número total de rondas requerido para llevar a cabo el protocolo incrementa con la proporción de relación. Cuando la proporción de relación es aumenta, las CAs deben esperar la actuación de un mayor número de vecinas antes de actuar en la segunda fase del protocolo.

También, el número de mensajes enviados por cada autoridad incrementa con la proporción de relación (Figura 7.9) ya que cada CA debe comunicar sus decisiones a todas sus vecinas. Así, un número grande de vecinas involucra el envío/recepción de más mensajes. La Figura 7.10 muestra la relación entre el número de mensajes enviados por cada CA y el número de vecinas por autoridad, a medida que se incrementa la proporción de relación. Como se observa, en todos los casos, el valor de esta relación es independiente del número de CAs, y en promedio una CA envía a cada una de sus vecinas 4 mensajes.

Las Figuras 7.8 y 7.9 son muy similares, aunque el número de mensajes enviados incrementa más rápido que el número de rondas, puesto que varias entidades pueden elegir su CA superior en la misma ronda.

Por otra parte, en la Figura 7.1 mostramos el número promedio de mensajes que firma cada CA cuando se lleva a cabo PROSEARCH. En esta figura se aprecia, que el número de mensajes firmados es independiente del número de CAs y varía de 4 a 6, por lo que el coste computacional de las operaciones criptográficas resultantes de la firma de estos mensajes es bajo para el verificador.

Las tablas de datos de las figuras mencionadas anteriormente se encuentran en el apéndice A, Tablas A.19, A.20, A.21 y A.22.

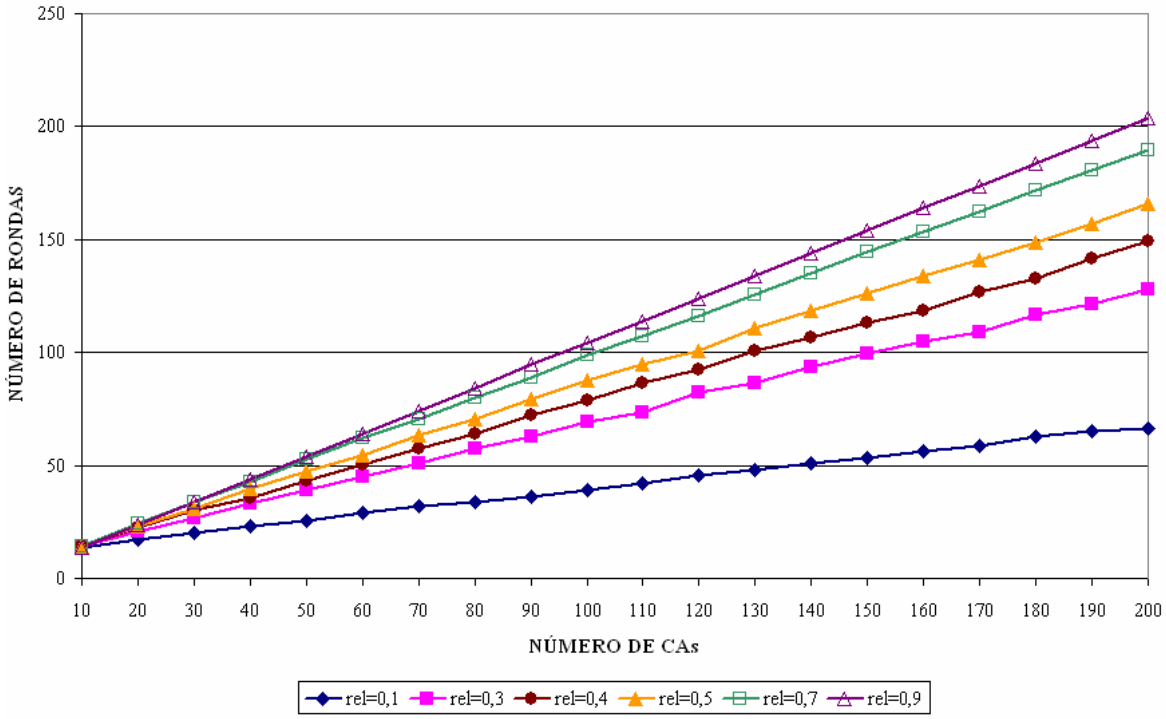

Figura 7.8: Número de CAs vs. número total de rondas

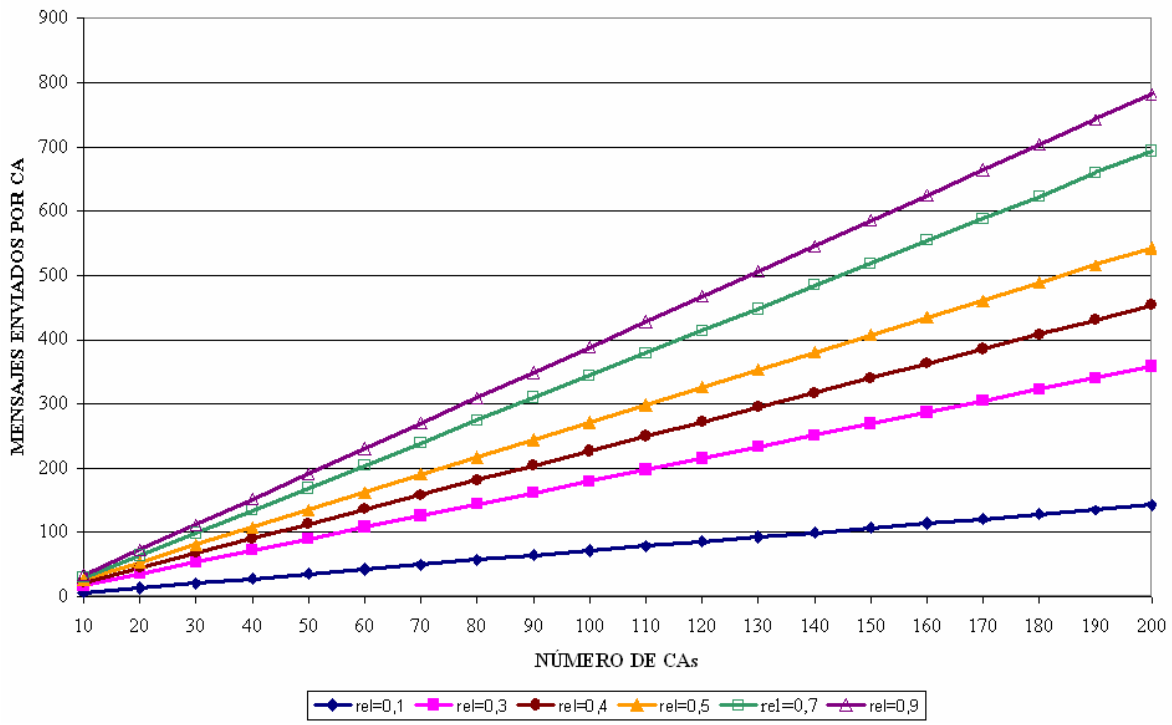

Figura 7.9: Número de CAs vs. mensajes enviados por cada CA

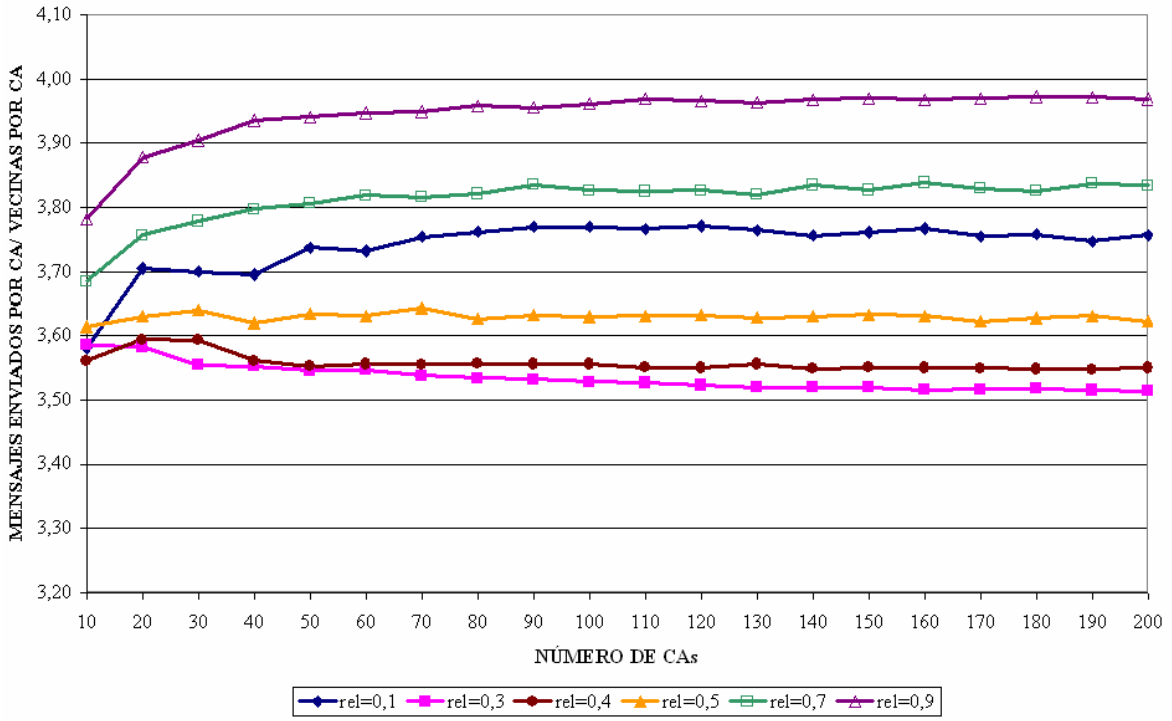

Figura 7.10: Relación entre el número de mensajes enviados y vecinas por cada CA

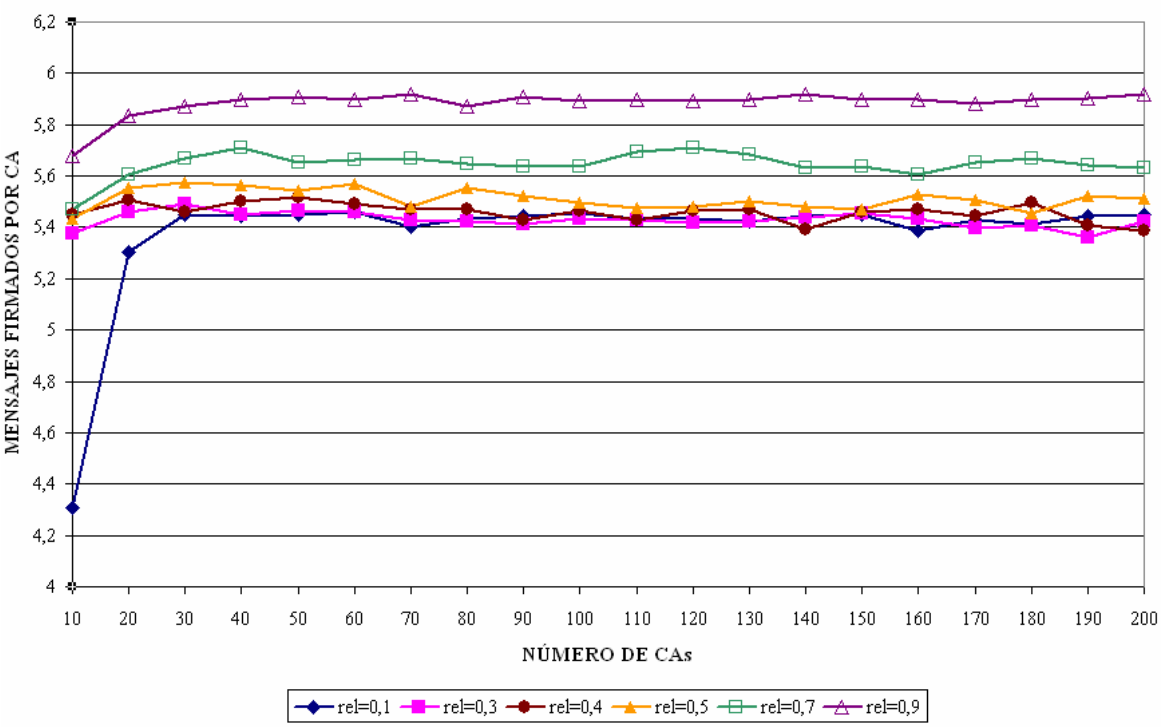

Figura 7.11: Número de CAs vs. mensajes firmados emitidos por cada CA

7.3.2.3 Variación de la Longitud Máxima de los Caminos

Ahora vamos a mantener la proporción de relación constante, rel=0,4, y a incrementar de2 a 7 la longitud máxima de los caminos. Como se observa en la Figura 7.12, el número de raíces se ve influenciado por L_{MAX} sólo cuando ésta es igual a 2, ya que al ser caminos muy cortos, nuestro protocolo no logra encontrar una sola raíz para la jerarquía. Esto se ve reflejado en la probabilidad de fallo (ver Figura 7.13), ya que la existencia de varias raíces hace que no se posible encontrar siempre un camino entre cada par de CAs. Las tablas de datos de estas figuras se encuentran en el apéndice A, Tablas A.23 y A.24.

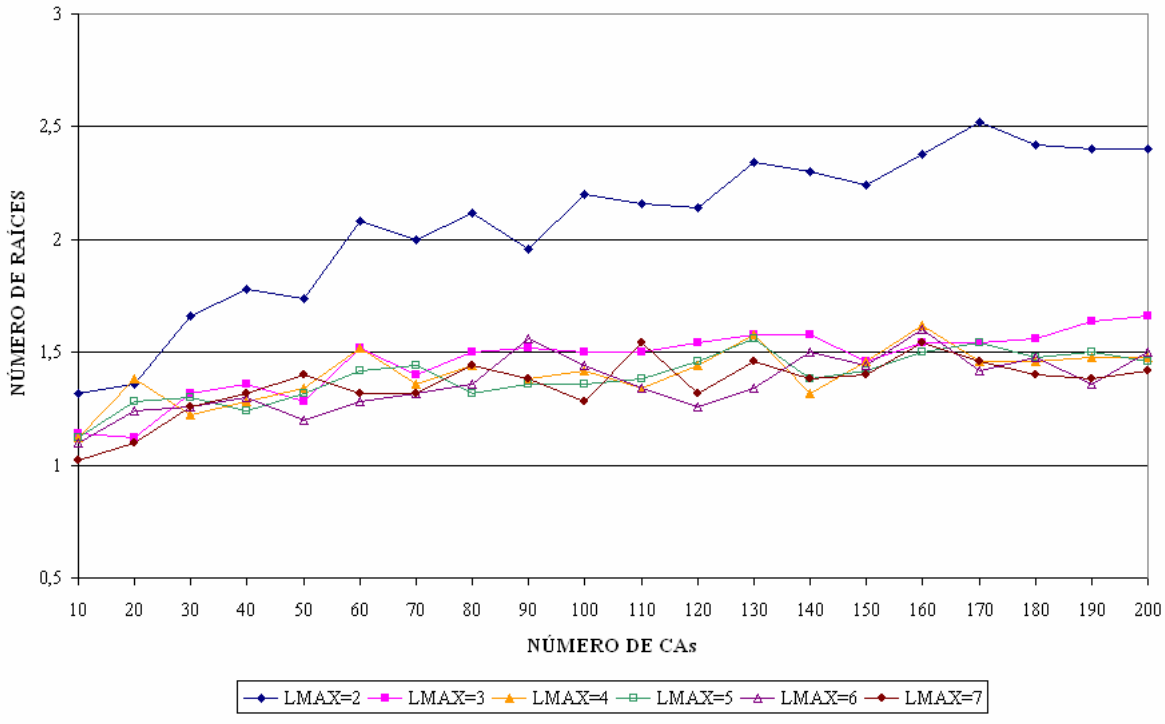

Figura 7.12: Número de CAs vs. número de raíces cuando varía L_{MAX}

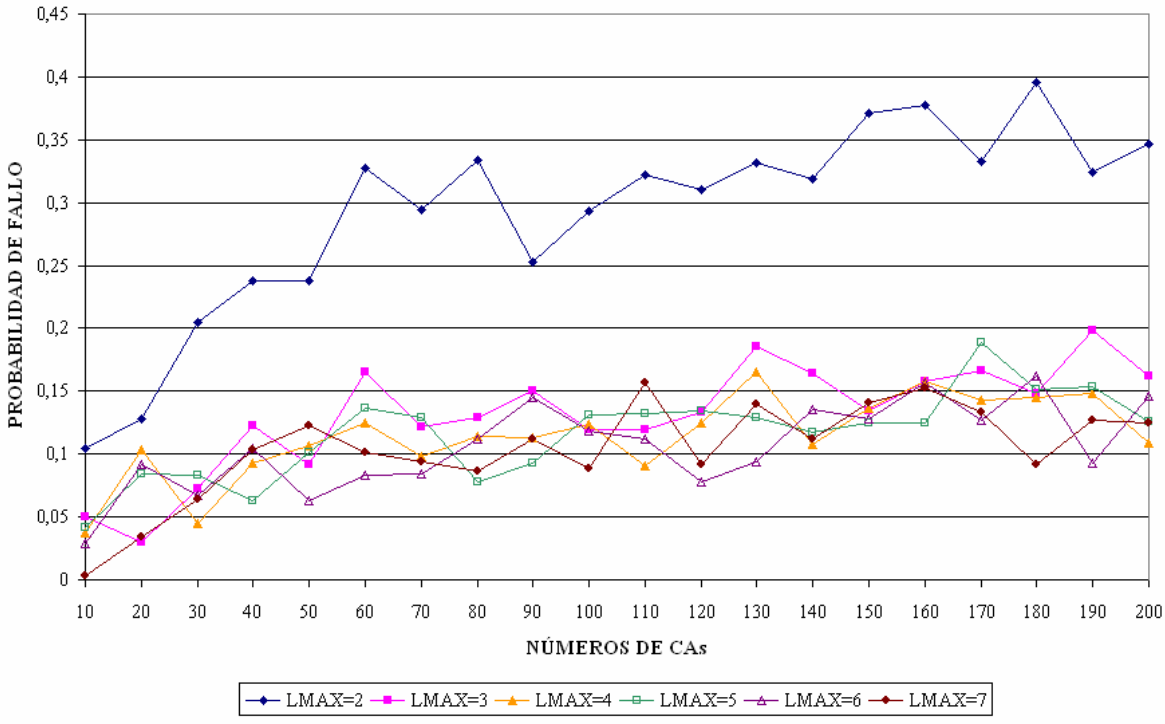

Figura 7.13: Número de CAs vs. probabilidad de fallo cuando varía L_{MAX}

Por otra parte, la Figura 7.14, muestra que aunque se incremente el valor de L_{MAX}, la longitud promedio de los caminos establecidos por nuestro protocolo va a ser menor que 2, por lo que el valor de la longitud máxima permitida no tiene gran influencia en la jerarquía establecida por PROSEARCH, gracias a que la elección de la vecina más confiable como CA superior garantiza caminos cortos. La tabla de datos de esta figura se encuentra en el apéndice A, Tabla A.25.

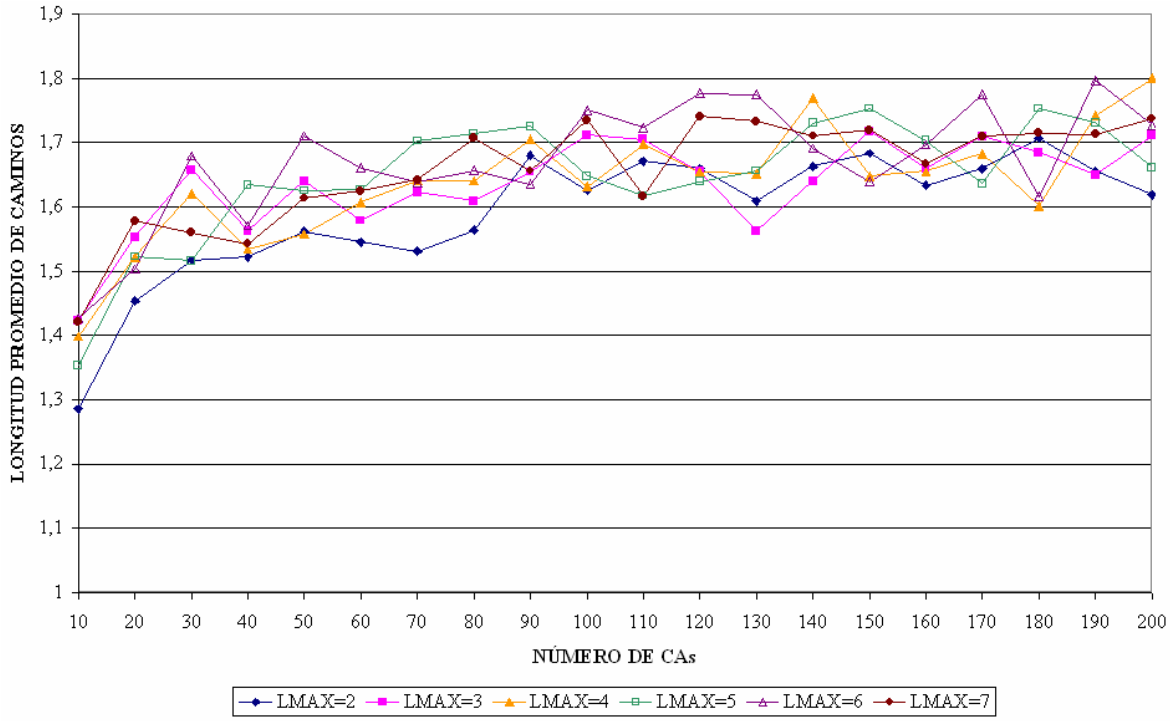

Figura 7.14: Número de CAs vs. longitud promedio de caminos cuando varía L_{MAX}

En cuanto al número de rondas y el número de mensajes enviados por cada autoridad, las Figuras 7.15 y 7.16, muestran que estos parámetros no se ven influenciados por L_{MAX}. Respecto al número de mensajes firmados por cada CA (ver Figura 7.17), cuando L_{MAX}=2, se requiere firmar un mensaje más cuando el número de autoridades es menor que 90, porque se requiere posiblemente la repetición del protocolo y es necesario firmar un mensaje de fallo para enviarlo a las vecinas. Las tablas de datos de estas figuras se encuentran en el apéndice A, Tabla A.26, A.27 y A.28.

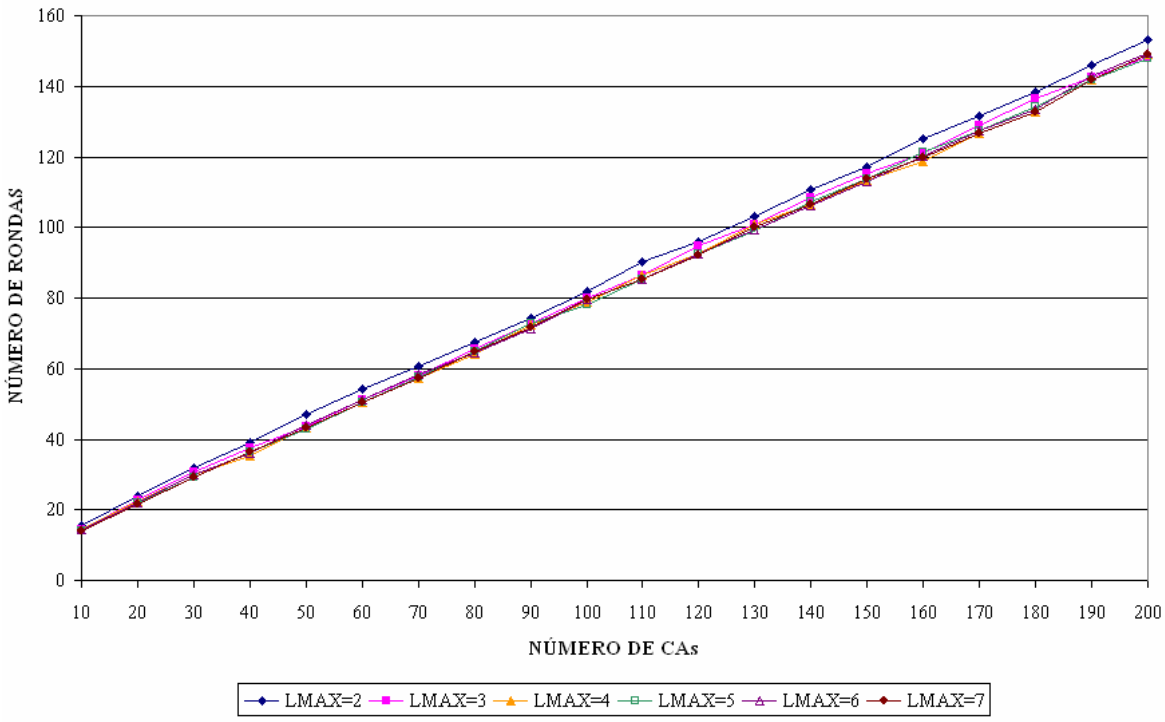

Figura 7.15: Número de CAs vs. número de rondas cuando varía L_{MAX}

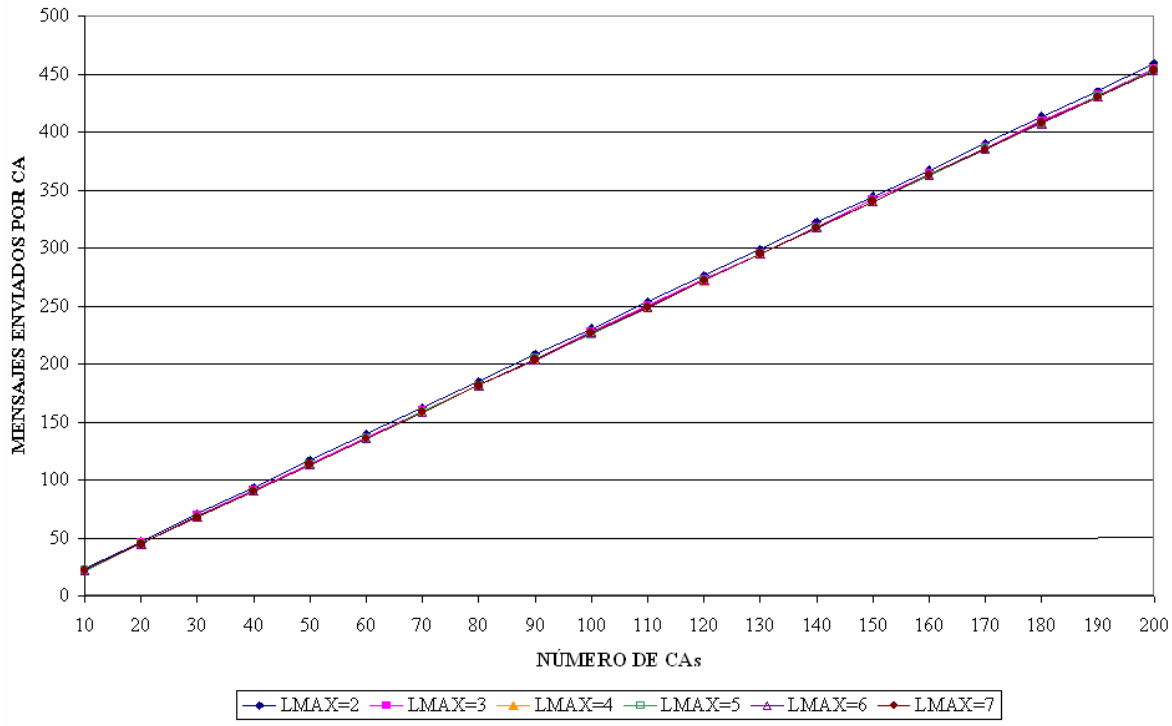

Figura 7.16: Número de CAs vs. número de mensajes enviados por CA cuando varía L_{MAX}

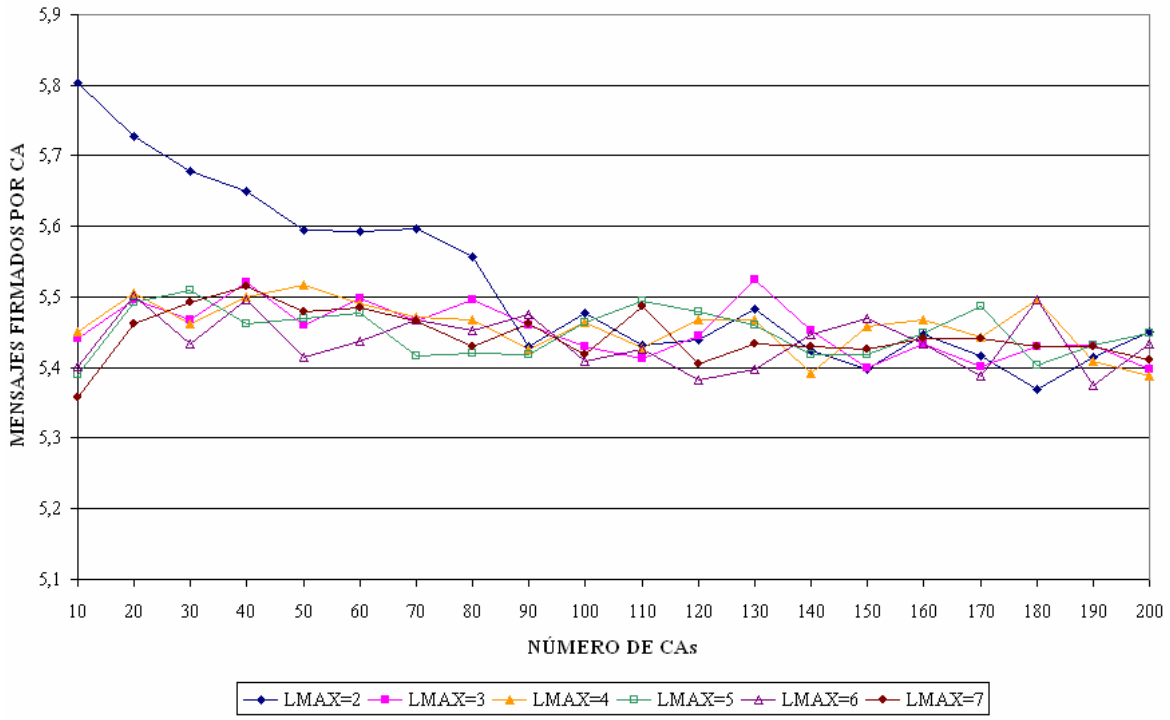

Figura 7.17: Número de CAs vs. número de mensajes firmados por CA cuando varía L_{MAX}

7.4 Conclusiones

La autenticación es un requerimiento importante de los sistemas de información críticos y PKI es ampliamente usada para proveer este servicio. Aunque las PKIs en malla son dinámicas y los caminos de certificación pueden construirse incluso si parte de la infraestructura no está disponible temporalmente, el descubrimiento de los caminos de certificación no es una tarea fácil debido a la existencia de relaciones de confianza bidireccionales. Éste no es el caso de las PKIs jerárquicas, donde existe un único camino entre dos entidades. Por tanto, PROSEARCH puede emplearse para simplificar el descubrimiento de los caminos de certificación en escenarios críticos. La Figura 7.1 muestra que el número de rondas necesarias para reconfigurar la jerarquía tras un desastre tiende a disminuir, ya que el tiempo de ejecución de nuestro protocolo depende directamente del número de CAs participantes. Esta reconfiguración rápida de PROSEARCH, lo hace apropiado para escenarios críticos. Adicionalmente, la Figura 7.2 muestra que el número de raíces tras la

reconfiguración es casi igual que el número de CAs raíz antes del desastre, por lo que la nueva jerarquía es tan funcional como la anterior.

El dinamismo de las redes ad-hoc móviles implica relaciones de confianza cambiantes entre sus nodos, así que las PKIs en malla también pueden aplicarse en este caso. Los resultados de nuestra simulación muestran que PROSEARCH es apropiado para PKIs con una proporción de relación mayor que 0,4. En estos casos, es muy probable encontrar una sola CA raíz y la probabilidad de que no se encuentre camino de una CA a otra es menor del 15% (ver Figuras 7.4 y 7.7). Cuando la proporción de relación es 0,3, se tienden a obtener dos raíces con una probabilidad de fallo hasta del 30%. Con una proporción de relación menor que 0,3 resultan varias CAs raíz y la probabilidad de fallo llega a ser algunas veces hasta del 70%. Por otra parte, el número de raíces resultantes y la probabilidad de fallo se ven poco influenciados por la longitud máxima de los caminos (ver Figuras 7.12 y 7.13). Sólo cuando la longitud máxima es igual a 2, el número de raíces y la probabilidad de fallo aumentan a medida que crece el número de CAs participantes, debido a que no todas las autoridades pueden hallar un camino tan corto a la CA más confiable. Para valores de L_{MAX} mayores que 2, el número de raíces y la probabilidad de fallo son bastante similares.

En la Figura 7.3 se puede ver la longitud máxima que alcanzan los caminos de certificación cuando se construye la mejor jerarquía posible entre las CAs. Como se aprecia, esta longitud máxima es siempre menor que 4 y con rel=0,1 tiende a 3 a medida que se incrementa el número de CAs. En los demás casos, la longitud máxima tiende a 2. A pesar de ello, nuestro protocolo no siempre establece la mejor jerarquía debido a que las CAs no tienen un conocimiento global de la PKI y sólo conocen su vecindario, por lo que hacer L_{MAX}=2 ocasionará múltiples raíces. Así, para obtener un mejor funcionamiento de PROSEARCH es conveniente que el valor de L_{MAX} sea mayor que 2, y según los resultados obtenidos no debería ser mayor que 4. Por otra parte, la Figura 7.5 muestra que la longitud promedio de los caminos de la jerarquía virtual establecida con PROSEARCH es menor que 2, lo que indica que la mayoría de los caminos establecidos son cortos.

Aunque, la jerarquía encontrada por nuestro protocolo no siempre es la mejor solución y no se garantizan los caminos más cortos pues se descartan algunas relaciones de confianza de

la PKI, en nuestra opinión, éste no es un problema importante para PROSEARCH pues los resultados de las simulaciones muestran que en la mayoría de los casos se encuentra una jerarquía aceptable, especialmente si consideramos la fácil implementación del protocolo y su simplicidad.

Por otra parte, el número de rondas y de mensajes enviados por cada CA incrementa linealmente con el número de CAs y la proporción de relación (ver Figuras 7.8 y 7.9), y es independiente de la longitud máxima de los caminos (ver Figuras 7.15 y 7.16).

El número de rondas aumenta con la proporción de relación porque las CAs deben esperar la actuación de un mayor número de vecinas antes de actuar en la segunda fase del protocolo y pocas autoridades actúan al mismo tiempo.

El número de mensajes enviados por cada autoridad incrementa con la proporción de relación porque cada CA debe comunicar sus decisiones a todas sus vecinas. Así, la cantidad de mensajes enviados está estrechamente relacionado con el número de vecinas como se aprecia en la Figura 7.10.

Por otra parte, el número de mensajes firmados por cada CA es independiente del número de CAs y de la longitud máxima de los caminos (ver Figuras 7.11 y 7.17). Cada CA máximo firma 6 mensajes, por lo que el coste computacional de las operaciones criptográficas involucradas en este proceso es bajo para dicha autoridad.

Capítulo 8

Conclusiones y Líneas Futuras

8.1 Conclusiones 165
8.2 Líneas Futuras 167

8.1 Conclusiones

En esta tesis hemos presentado el estado del arte tanto de los mecanismos de autenticación y autorización, como del proceso de validación de caminos. La validación es parte fundamental de los servicios de autenticación y autorización, porque permite constatar la validez e integridad de la información que se comparte. En las arquitecturas basadas en certificados, como PKI y PMI, el proceso de validación consiste básicamente en verificar la firma y vigencia de los certificados.

Cuando el usuario desconoce la clave pública del emisor de un certificado y quiere verificar la firma de éste, debe formar una cadena de certificados desde su base de confianza hasta el propietario del certificado, llamada en PKI camino de certificación.

Uno de los objetivos de este trabajo ha sido determinar el coste computacional y la capacidad de almacenamiento demandados por el proceso de validación de caminos de certificación y de delegación a un verificador, cuando se utilizan los mecanismos de revocación estándar. De este estudio podemos concluir que cuando se utilizan CRLs de gran tamaño, conteniendo la información de revocación de por lo menos 1000 certificados, el verificador requiere gran capacidad de almacenamiento y de procesamiento para llevar a cabo la validación de caminos. Por tal motivo, cuando el número de certificados revocados es grande, OCSP es el mecanismo de revocación más apropiado para verificadores con

capacidades limitadas. Aún así, el coste computacional y la capacidad de almacenamiento son parámetros críticos para dispositivos como los teléfonos móviles y las tarjetas inteligentes.

El proceso de validación de caminos de certificación es bastante complejo, pues involucra: descubrir los caminos, recuperar los certificados que hacen parte de ellos, verificar la firma digital de cada uno de estos certificados, y constatar que ninguno haya expirado o haya sido revocado. Dada la complejidad de este proceso, el objetivo principal del presente trabajo de tesis ha sido proponer diferentes soluciones que conduzcan a simplificarlo desde la perspectiva del verificador. Por tal motivo, hemos propuesto dos mecanismos que contribuyen a simplificar diferentes aspectos del proceso de validación.

En primer lugar, proponemos TRUTHC, un mecanismo alternativo de validación de cadenas de certificados que permite verificar su integridad y constatar que cada uno de ellos pertenece al mismo camino de certificación reemplazando las operaciones de verificación de firma por operaciones de hash.

TRUTHC establece otro tipo de relación de confianza entre las entidades de las PKIs jerárquicas utilizando dos cadenas de hash: una encadena las semillas secretas de las CAs y la otra encadena el contenido de los certificados de cada camino de certificación. Esto permite reducir el coste computacional del proceso de validación aunque incrementa un poco el coste del proceso de expedición de certificados para las CAs subordinadas. TRUTHC es compatible con los certificados X.509 y su seguridad depende en gran medida de la confidencialidad de las semillas.

TRUTHC puede aplicarse en escenarios donde las entidades que quieren verificar la validez de los certificados tienen capacidad de procesamiento limitada y deben delegar el proceso de validación en otra entidad, es el caso de las redes móviles que disponen de servidores de validación.

En segundo lugar, proponemos PROSEARCH, un protocolo que permite simplificar la construcción de los caminos de certificación en PKIs donde las relaciones de confianza se han generado dinámicamente según un modelo peer-to-peer. Este protocolo establece una jerarquía virtual basado en el nivel de confiabilidad de las CAs participantes, que se determina utilizando dos parámetros: el número de autoridades en que confía una CA (IN_i) y el número

de entidades que confían en ella (OUT_i). Los resultados obtenidos demuestran que PROSEARCH es apropiado para PKIs con una proporción de relación entre sus entidades superior al 40%, ya que en estos casos, es muy probable encontrar una sola CA raíz y la probabilidad de que no se encuentre camino de una CA a otra es menor del 15%. Además, la longitud máxima de los caminos debe ser de 3 o 4 certificados, para que la jerarquía establecida sea lo más parecida posible a la mejor solución.

Aunque, la jerarquía encontrada por nuestro protocolo no siempre es la mejor solución, debido a que las CAs no tienen un conocimiento global de la arquitectura, y no se garantizan los caminos más cortos, pues se descartan algunas relaciones de confianza de la PKI, en nuestra opinión, éste no es un problema importante para PROSEARCH pues los resultados de las simulaciones muestran que en la mayoría de los casos se encuentra una jerarquía aceptable, especialmente si consideramos la fácil implementación del protocolo y su simplicidad. Además, PROSEARCH requiere poca intervención por parte del usuario. Estas propiedades unidas a su rápida ejecución hacen a nuestro protocolo aplicable en redes ad-hoc móviles, que son redes altamente dinámicas con relaciones de confianza cambiantes entre sus nodos.

La reconfiguración rápida de PROSEARCH, lo hace también apropiado para escenarios críticos. Los resultados muestran que el número de rondas necesarias para reconfigurar la jerarquía tras un desastre tiende a disminuir, ya que el tiempo de ejecución de nuestro protocolo depende directamente del número de CAs participantes. Adicionalmente, el número de raíces tras la reconfiguración es casi igual que el número de CAs raíz antes del desastre, por lo que la nueva jerarquía es tan funcional como la anterior.

8.2 Líneas Futuras

Durante la realización de este trabajo han surgido diferentes temas que no han podido ser abarcados y nuevas ideas orientadas a la aplicación de las soluciones propuestas en diferentes escenarios.

En el caso de TRUTHC, se requiere crear un mecanismo eficiente para actualizar las semillas secretas. Otra posible línea futura es la evaluación de la escalabilidad del sistema al incrementar el número de VAs y la eficacia del proceso comunicativo entre las diferentes entidades. Además, se pueden estudiar las implicaciones del uso de TRUTHC en PMI, ya que sólo hemos presentado la idea de como se podría hacer.

Por otra parte, los resultados obtenidos con PROSEARCH nos muestran que la jerarquía establecida no siempre es la mejor solución, por lo que un estudio analítico de la teoría de grafos y algoritmos de búsqueda de caminos pueden llevarnos a introducir modificaciones en nuestro protocolo para obtener mejores resultados. También se puede realizar un estudio estadístico que nos permita fijar el valor de $L_{MAX,}$ ya que hasta el momento es la autoridad que inicia el protocolo la que decide que valor darle a este parámetro.

Un aspecto surgido al evaluar la seguridad de PROSEARCH es la inexistencia de un mecanismo que permita identificar con certeza cuando una entidad es maliciosa. Por lo que una línea futura es el estudio de las diferentes técnicas de detección de miembros deshonestos y la evaluación de su aplicabilidad en PROSEARCH.

Además, hemos supuesto en nuestro protocolo, que los mensajes que envían las entidades a sus vecinas llegan al destino, pero en realidad no se tiene seguridad de ello, ya que se emplearían mensajes multicast, cuyo transporte, por definición, no es fiable. Éste sería por tanto otro aspecto a mejorar en PROSEARCH.

Finalmente, pensamos que se podrían establecer diferentes jerarquías de acuerdo al nivel de seguridad requerido por las autoridades, lo que permitiría la aplicación de PROSEARCH en otros escenarios.

Apéndice A

Tablas de Datos

Este apéndice contiene las tablas con los resultados obtenidos durante nuestra investigación para los diferentes casos analizados.

Tabla A.1: Coste computacional sin y con revocación usando RSA

LONGITUD DEL CAMINO (L_c)	COSTE COMPUTACIONAL (ms)				
	SIN REVOCACIÓN	CON CRL		CON OCSP	
		R_c=0	R_c=L_c	R_c=1	R_c=L_c
1	5,20	5,20	10,40	10,78	10,78
2	10,40	10,40	20,80	16,36	21,56
3	15,60	15,60	31,20	21,94	32,34
4	20,80	20,80	41,60	27,52	43,12
5	26,00	26,00	52,00	33,10	53,90
6	31,20	31,20	62,40	38,68	64,68
7	36,40	36,40	72,80	44,26	75,46
8	41,60	41,60	83,20	49,84	86,24
9	46,80	46,80	93,60	55,42	97,02
10	52,00	52,00	104,00	61,00	107,80

Tabla A.2: Coste computacional sin y con revocación usando ECDSA

LONGITUD DEL CAMINO (L_c)	COSTE COMPUTACIONAL (ms)				
	SIN REVOCACIÓN	CON CRL		CON OCSP	
		R_c=0	R_c=L_c	R_c=1	R_c=L_c
1	46,69	46,69	93,38	93,76	93,76
2	93,38	93,38	186,76	140,83	187,52
3	140,07	140,07	280,14	187,90	281,28
4	186,76	186,76	373,52	234,97	375,04
5	233,45	233,45	466,90	282,04	468,80
6	280,14	280,14	560,28	329,11	562,56
7	326,83	326,83	653,66	376,18	656,32
8	373,52	373,52	747,04	423,25	750,08
9	420,21	420,21	840,42	470,32	843,84
10	466,90	466,90	933,80	517,39	937,60

Tabla A.3: Coste computacional con RSA (1024 bits) y ECDSA (163 bits) para PDA con procesador StrongARM a 200MHz

MECANISMO DE REVOCACIÓN (L_c=10)	COSTE COMPUTACIONAL (ms)	
	RSA	ECDSA
SIN REVOCACIÓN	52,00	466,90
CRL (R_c=0)	52,00	466,90
CRL (R_c=L_c)	104,00	933,80
OCSP(R_c=1)	60,90	517,39
OCSP(R_c=L_c)	107,70	937,60

Tabla A.4: Coste computacional sin y con revocación de caminos de delegación

L_d	COSTE COMPUTACIONAL (ms)					
	SIN REV.	CON CRL			CON OCSP	
		R_c=R_d=0	R_c=L_c, R_d=L_d, N=500	R_c=L_c, R_d=L_d, N=10^4	R_c=R_d=1	R_c=L_c, R_d=L_d
1	0,75	0,75	1,56	3,26	1,11	1,47
2	1,50	1,50	3,12	6,52	2,04	2,94
3	2,25	2,25	4,68	9,78	2,97	4,41
4	3,00	3,00	6,24	13,04	3,90	5,88
5	3,75	3,75	7,80	16,30	4,83	7,35
6	4,50	4,50	9,36	19,56	5,76	8,82
7	5,25	5,25	10,92	22,82	6,69	10,29
8	6,00	6,00	12,48	26,08	7,62	11,76
9	6,75	6,75	14,04	29,34	8,55	13,23
10	7,50	7,50	15,60	32,60	9,48	14,70

Tabla A.5: Capacidad de almacenamiento sin y con revocación usando RSA (1024 bits)

L_c	CAPACIDAD DE ALMACENAMIENTO (KB)				
	SIN REVOCACIÓN	CON CRL		CON OCSP	
		R_c=1, *RCert*=100	R_c=L_c, *RCert*=100	R_c=1	R_c=L_c
1	1,29	4,65	4,65	1,60	1,60
2	1,75	5,11	8,46	2,19	2,38
3	2,22	5,57	12,28	2,77	3,15
4	2,68	6,03	16,10	3,35	3,93
5	3,14	6,50	19,92	3,93	4,70
6	3,60	6,96	23,73	4,51	5,48
7	4,06	7,42	27,55	5,09	6,25
8	4,53	7,88	31,37	5,67	7,03
9	4,99	8,34	35,19	6,25	7,80
10	5,45	8,80	39,00	6,83	8,57

Tabla A.6: Capacidad de almacenamiento sin y con revocación usando ECDSA (163 bits)

L_c	CAPACIDAD DE ALMACENAMIENTO (KB)				
	SIN REVOCACIÓN	CON CRL		CON OCSP	
		R_c=1, *RCert*=100	R_c=L_c, *RCert*=100	R_c=1	R_c=L_c
1	0,67	3,92	3,92	0,87	0,87
2	0,92	4,17	7,42	1,25	1,33
3	1,17	4,42	10,92	1,62	1,79
4	1,42	4,67	14,43	1,99	2,26
5	1,68	4,93	17,93	2,36	2,72
6	1,93	5,18	21,44	2,73	3,18
7	2,18	5,43	24,94	3,11	3,64
8	2,44	5,69	28,44	3,48	4,10
9	2,69	5,94	31,95	3,85	4,56
10	2,94	6,19	35,45	4,22	5,02

Tabla A.7: Capacidad de almacenamiento con CRL usando RSA (1024 bits)

	CAPACIDAD DE ALMACENAMIENTO (KB)			
L_c	R_c=1, *RCert*=100	R_c=L_c, *RCert*=100	R_c=1, *RCert*=1000	R_c=L_c, *RCert*=1000
1	4,65	4,65	32,77	32,77
2	5,11	8,46	33,23	64,71
3	5,57	12,28	33,70	96,66
4	6,03	16,10	34,16	128,60
5	6,50	19,92	34,62	160,54
6	6,96	23,73	35,08	192,48
7	7,42	27,55	35,54	224,43
8	7,88	31,37	36,01	256,37
9	8,34	35,19	36,47	288,31
10	8,80	39,00	36,93	320,25

Tabla A.8: Capacidad de almacenamiento con CRL usando ECDSA (163 bits)

	CAPACIDAD DE ALMACENAMIENTO (KB)			
L_c	R_c=1, *RCert*=100	R_c=L_c, *RCert*=100	R_c=1, *RCert*=1000	R_c=L_c, *RCert*=1000
1	3,92	3,92	32,04	32,04
2	4,17	7,42	32,29	63,67
3	4,42	10,92	32,55	95,30
4	4,67	14,43	32,80	126,93
5	4,93	17,93	33,05	158,56
6	5,18	21,44	33,31	190,19
7	5,43	24,94	33,56	221,81
8	5,69	28,44	33,81	253,44
9	5,94	31,95	34,06	285,07
10	6,19	35,45	34,32	316,70

Tabla A.9: Capacidad de almacenamiento con RSA (1024 bits) y ECDSA (163 bits)

MECANISMO DE REVOCACIÓN (L_c=10)	CAPACIDAD DE ALMACENAMIENTO (KB)	
	RSA	ECDSA
SIN REVOCACIÓN	5,45	2,94
CRL (R_c=1, *RCert*=100)	8,80	6,19
CRL (R_c=1, *RCert*=1000)	36,93	34,32
CRL (R_c=L_c, *RCert*=100)	39,00	35,45
OCSP(R_c=1)	6,83	4,22
OCSP(R_c=L_c)	8,57	5,02

Tabla A.10: Capacidad de almacenamiento en ejemplo de aplicación

MECANISMO DE REVOCACIÓN (L_c=10)	CAPACIDAD DE ALMACENAMIENTO (KB)	
	RSA	ECDSA
SIN REVOCACIÓN	2,59	1,25
CRL (*RCert*=500)	34,43	32,76
OCSP	3,46	1,79

Tabla A.11: Coste computacional de TRUTHC vs. PKI típica

LONGITUD DEL CAMINO (L_c)	VERIFICADOR (PKI TÍPICA)	VA (PKI CON TRUTHC)
1	5,20	9,53
2	10,40	9,53
3	15,60	9,54
4	20,80	9,54
5	26,00	9,55
6	31,20	9,55
7	36,40	9,56
8	41,60	9,56
9	46,80	9,57
10	52,00	9,57

Tabla A.12: Número de CAs vs. número de rondas

CAs	NÚMERO DE RONDAS					
	rel=0,2		rel=0,4		rel=0,6	
	SIN FALLOS	CON FALLOS	SIN FALLOS	CON FALLOS	SIN FALLOS	CON FALLOS
10	14,06	11,46	14,80	11,38	14,42	11,02
20	19,14	16,14	22,56	17,26	23,84	17,88
30	23,92	20,20	30,38	23,24	33,58	24,70
40	28,82	23,30	37,58	28,66	42,36	30,76
50	34,32	26,56	45,18	34,04	50,42	36,64
60	37,94	29,40	51,20	38,64	58,96	43,18
70	43,08	33,10	58,76	43,44	68,38	49,82
80	47,90	35,76	65,96	48,52	76,66	56,02
90	52,00	39,80	72,52	53,48	83,62	62,34
100	56,04	42,54	79,06	59,06	92,94	67,90
110	61,52	46,40	86,42	63,76	102,64	73,76
120	66,32	49,76	92,92	67,96	110,68	79,66
130	70,40	52,58	101,60	72,82	119,56	85,44
140	75,80	55,70	106,74	78,40	128,06	91,22
150	79,06	58,34	114,58	82,14	136,18	98,18
160	84,22	62,26	122,30	87,42	144,90	104,08
170	88,74	65,40	127,62	92,46	154,48	109,52
180	91,44	68,32	135,02	98,68	162,38	116,00
190	97,68	71,98	142,46	102,82	171,56	122,46
200	102,24	75,40	149,68	108,38	179,70	129,28

Tabla A.13: Número de CAs vs. número de CAs raíz

CAs	NÚMERO DE CAs RAÍZ					
	rel=0,2		rel=0,4		rel=0,6	
	SIN FALLOS	CON FALLOS	SIN FALLOS	CON FALLOS	SIN FALLOS	CON FALLOS
10	1,72	1,62	1,12	1,08	1,02	1,04
20	2,06	2,02	1,26	1,16	1,06	1,02
30	2,28	2,04	1,32	1,26	1,12	1,06
40	2,38	2,18	1,40	1,28	1,12	1,08
50	2,60	2,40	1,34	1,40	1,14	1,12
60	2,74	2,52	1,38	1,40	1,06	1,16
70	2,80	2,52	1,46	1,34	1,04	1,10
80	2,90	2,56	1,54	1,36	1,16	1,20
90	2,92	2,64	1,36	1,48	1,14	1,08
100	2,94	2,62	1,56	1,36	1,16	1,08
110	2,88	2,68	1,44	1,42	1,10	1,08
120	2,80	3,04	1,58	1,46	1,16	1,06
130	3,06	3,00	1,54	1,44	1,18	1,12
140	3,08	2,74	1,52	1,34	1,24	1,12
150	2,98	2,92	1,66	1,58	1,08	1,04
160	2,96	2,82	1,58	1,54	1,12	1,20
170	2,98	3,24	1,50	1,64	1,14	1,14
180	3,08	2,92	1,64	1,64	1,16	1,10
190	3,34	3,32	1,44	1,60	1,10	1,12
200	3,22	3,20	1,62	1,54	1,22	1,08

Tabla A.14: Longitud máxima de los caminos con mejor jerarquía

CAs	LONGITUD MÁXIMA DE CAMINOS CON MEJOR JERARQUÍA					
	rel=0,1	rel=0,3	rel=0,4	rel=0,5	rel=0,7	rel=0,9
10	3,74	2,06	2,00	1,98	1,68	1,00
20	3,90	2,02	2,00	2,00	2,00	1,06
30	3,36	2,00	2,00	2,00	2,00	1,22
40	3,12	2,00	2,00	2,00	2,00	1,58
50	3,04	2,00	2,00	2,00	2,00	1,82
60	3,00	2,00	2,00	2,00	2,00	1,94
70	3,02	2,00	2,00	2,00	2,00	2,00
80	3,00	2,00	2,00	2,00	2,00	1,98
90	3,00	2,00	2,00	2,00	2,00	2,00
100	3,00	2,00	2,00	2,00	2,00	2,00
110	3,00	2,00	2,00	2,00	2,00	2,00
120	3,00	2,00	2,00	2,00	2,00	2,00
130	3,00	2,00	2,00	2,00	2,00	2,00
140	3,00	2,00	2,00	2,00	2,00	2,00
150	3,00	2,00	2,00	2,00	2,00	2,00
160	2,96	2,00	2,00	2,00	2,00	2,00
170	2,98	2,00	2,00	2,00	2,00	2,00
180	2,98	2,00	2,00	2,00	2,00	2,00
190	3,00	2,00	2,00	2,00	2,00	2,00
200	2,94	2,00	2,00	2,00	2,00	2,00

Tabla A.15: Número de CAs vs. número promedio de raíces

CAs	NÚMERO DE RAÍCES					
	rel=0,1	rel=0,3	rel=0,4	rel=0,5	rel=0,7	rel=0,9
10	2,12	1,32	1,12	1,06	1,02	1,00
20	3,00	1,44	1,38	1,10	1,00	1,00
30	3,26	1,54	1,22	1,26	1,06	1,00
40	3,62	1,56	1,28	1,10	1,04	1,00
50	3,86	1,68	1,34	1,24	1,04	1,00
60	3,94	1,88	1,52	1,22	1,02	1,00
70	3,84	1,64	1,36	1,14	1,02	1,00
80	4,12	1,70	1,44	1,28	1,04	1,00
90	4,52	1,88	1,38	1,22	1,06	1,00
100	4,32	1,84	1,42	1,18	1,02	1,00
110	4,62	2,00	1,34	1,18	1,06	1,00
120	4,36	1,80	1,44	1,24	1,12	1,00
130	4,60	1,86	1,58	1,28	1,04	1,00
140	4,68	1,90	1,32	1,24	1,04	1,02
150	4,84	2,10	1,46	1,24	1,06	1,02
160	4,68	1,86	1,62	1,28	1,04	1,00
170	4,92	1,90	1,46	1,22	1,02	1,00
180	4,86	1,98	1,46	1,14	1,06	1,00
190	5,24	1,68	1,48	1,38	1,06	1,00
200	5,50	1,86	1,48	1,28	1,08	1,00

Tabla A.16: Número de CAs vs. longitud promedio de los caminos

CAs	LONGITUD PROMEDIO DE CAMINOS					
	rel=0,1	**rel=0,3**	**rel=0,4**	**rel=0,5**	**rel=0,7**	**rel=0,9**
10	1,67	1,55	1,39	1,29	1,19	1,00
20	1,72	1,67	1,52	1,37	1,26	1,00
30	1,81	1,65	1,62	1,41	1,24	1,00
40	1,81	1,71	1,53	1,44	1,23	1,01
50	1,82	1,71	1,55	1,50	1,31	1,01
60	1,85	1,69	1,60	1,44	1,31	1,04
70	1,96	1,76	1,64	1,57	1,31	1,02
80	1,86	1,78	1,64	1,48	1,34	1,08
90	1,92	1,77	1,70	1,54	1,36	1,04
100	1,87	1,75	1,63	1,58	1,35	1,06
110	1,89	1,75	1,69	1,60	1,29	1,07
120	1,89	1,76	1,65	1,58	1,29	1,07
130	1,90	1,77	1,65	1,57	1,32	1,07
140	1,86	1,71	1,76	1,58	1,37	1,05
150	1,84	1,68	1,64	1,59	1,36	1,07
160	1,95	1,71	1,65	1,55	1,40	1,07
170	1,89	1,77	1,68	1,54	1,35	1,09
180	1,90	1,75	1,60	1,63	1,33	1,08
190	1,83	1,82	1,74	1,56	1,36	1,08
200	1,83	1,75	1,79	1,56	1,38	1,06

Tabla A.17: Longitud promedio de caminos con mejor jerarquía

CAs	LONGITUD PROMEDIO DE CAMINOS CON MEJOR JERARQUÍA					
	rel=0,1	rel=0,3	rel=0,4	rel=0,5	rel=0,7	rel=0,9
10	1,82	1,26	1,17	1,07	0,99	0,90
20	2,15	1,40	1,29	1,18	1,07	0,95
30	2,07	1,46	1,34	1,24	1,11	0,97
40	2,00	1,49	1,37	1,27	1,12	0,98
50	2,00	1,51	1,40	1,30	1,14	0,99
60	1,96	1,52	1,41	1,32	1,14	1,00
70	1,93	1,53	1,42	1,32	1,16	1,00
80	1,90	1,54	1,44	1,33	1,16	1,01
90	1,90	1,55	1,45	1,34	1,17	1,01
100	1,88	1,56	1,45	1,35	1,17	1,02
110	1,89	1,56	1,46	1,36	1,18	1,02
120	1,87	1,57	1,46	1,37	1,18	1,02
130	1,86	1,57	1,47	1,37	1,18	1,03
140	1,86	1,58	1,47	1,37	1,19	1,03
150	1,85	1,58	1,48	1,38	1,20	1,03
160	1,85	1,58	1,48	1,38	1,19	1,03
170	1,85	1,59	1,48	1,38	1,20	1,03
180	1,84	1,59	1,48	1,38	1,20	1,03
190	1,84	1,59	1,49	1,39	1,20	1,04
200	1,85	1,59	1,49	1,39	1,21	1,03

Tabla A.18: Número de CAs vs. probabilidad de fallo

CAs	PROBABILIDAD DE FALLO					
	rel=0,1	**rel=0,3**	**rel=0,4**	**rel=0,5**	**rel=0,7**	**rel=0,9**
10	0,39	0,10	0,03	0,01	0,00	0,00
20	0,50	0,12	0,10	0,02	0,00	0,00
30	0,52	0,17	0,04	0,06	0,01	0,00
40	0,54	0,18	0,09	0,02	0,00	0,00
50	0,56	0,20	0,10	0,06	0,01	0,00
60	0,57	0,23	0,12	0,06	0,00	0,00
70	0,54	0,20	0,09	0,05	0,00	0,00
80	0,61	0,20	0,11	0,08	0,01	0,00
90	0,60	0,21	0,11	0,05	0,01	0,00
100	0,59	0,23	0,12	0,04	0,00	0,00
110	0,59	0,24	0,09	0,04	0,00	0,00
120	0,58	0,21	0,12	0,07	0,03	0,00
130	0,58	0,25	0,16	0,08	0,01	0,00
140	0,61	0,23	0,10	0,06	0,00	0,00
150	0,62	0,32	0,13	0,05	0,01	0,00
160	0,60	0,24	0,15	0,10	0,00	0,00
170	0,62	0,23	0,14	0,07	0,00	0,00
180	0,63	0,26	0,14	0,03	0,02	0,00
190	0,67	0,20	0,14	0,08	0,01	0,00
200	0,68	0,23	0,10	0,06	0,02	0,00

Tabla A.19: Número de CAs vs. número total de rondas

CAs	NÚMERO DE RONDAS					
	rel=0,1	**rel=0,3**	**rel=0,4**	**rel=0,5**	**rel=0,7**	**rel=0,9**
10	13,88	14,02	14,22	14,26	14,26	13,76
20	17,42	20,72	22,22	22,92	24,04	23,76
30	20,22	26,64	29,92	30,86	33,66	33,82
40	23,20	32,88	35,44	39,54	42,68	44,04
50	25,34	39,32	43,38	47,14	52,7	54,20
60	28,94	45,00	50,32	54,42	62,00	64,22
70	31,86	51,14	57,24	63,52	70,60	74,32
80	33,60	57,44	64,10	70,28	80,14	84,18
90	36,06	62,56	72,32	79,28	88,74	94,54
100	39,36	69,40	78,86	87,60	99,00	104,18
110	41,90	73,68	86,50	94,78	107,50	114,02
120	45,40	82,14	92,36	100,86	116,38	124,00
130	48,12	86,22	100,68	110,60	125,66	133,88
140	50,94	93,34	106,38	118,44	135,20	143,98
150	53,34	99,46	113,22	126,48	144,42	154,02
160	56,08	104,72	118,64	133,76	153,48	163,96
170	58,78	109,06	126,54	140,84	162,08	173,80
180	62,84	116,98	132,62	148,90	171,52	183,70
190	65,04	121,34	141,72	156,84	180,70	193,88
200	66,50	127,86	149,18	166,14	189,74	203,76

Tabla A.20: Número de CAs vs. mensajes enviados por cada CA

CAs	MENSAJES ENVIADOS POR CA					
	rel=0,1	**rel=0,3**	**rel=0,4**	**rel=0,5**	**rel=0,7**	**rel=0,9**
10	6,77	17,74	22,11	26,26	29,59	33,64
20	13,93	35,80	45,43	53,26	64,19	72,89
30	21,04	54,04	68,64	80,68	98,82	112,04
40	27,97	72,25	90,61	108,00	134,35	151,86
50	35,43	90,17	113,44	135,47	168,99	191,09
60	42,61	108,34	136,20	162,42	204,20	230,45
70	49,92	126,13	158,60	190,17	239,21	269,70
80	57,17	144,07	181,74	216,80	274,13	309,40
90	64,51	161,59	204,26	244,06	310,02	348,49
100	71,66	179,49	226,69	271,54	344,33	388,20
110	78,74	197,64	249,64	298,48	379,09	428,24
120	85,94	215,40	271,89	326,05	413,75	467,18
130	92,87	232,98	295,38	352,83	447,78	506,08
140	99,90	251,00	317,38	379,98	484,40	545,98
150	107,11	269,17	340,43	407,79	518,35	585,54
160	114,39	286,45	363,08	434,68	555,02	624,51
170	121,23	304,55	385,54	460,60	588,19	664,19
180	128,49	322,76	408,14	488,95	622,44	703,84
190	135,25	340,54	430,61	516,32	659,60	743,14
200	142,61	358,37	453,99	542,21	693,33	781,68

Tabla A.21: Relación entre el número de mensajes enviados y vecinas por cada CA

CAs	MENSAJES ENVIADOS POR CA/VECINAS POR CA					
	rel=0,1	rel=0,3	rel=0,4	rel=0,5	rel=0,7	rel=0,9
10	3,58	3,59	3,56	3,61	3,69	3,78
20	3,70	3,58	3,59	3,63	3,76	3,88
30	3,70	3,56	3,59	3,64	3,78	3,90
40	3,70	3,55	3,56	3,62	3,80	3,94
50	3,74	3,55	3,55	3,63	3,81	3,94
60	3,73	3,55	3,56	3,63	3,82	3,95
70	3,75	3,54	3,56	3,64	3,82	3,95
80	3,76	3,53	3,56	3,63	3,82	3,96
90	3,77	3,53	3,56	3,63	3,84	3,96
100	3,77	3,53	3,56	3,63	3,83	3,96
110	3,77	3,53	3,55	3,63	3,83	3,97
120	3,77	3,52	3,55	3,63	3,83	3,97
130	3,76	3,52	3,56	3,63	3,82	3,96
140	3,76	3,52	3,55	3,63	3,83	3,97
150	3,76	3,52	3,55	3,63	3,83	3,97
160	3,77	3,52	3,55	3,63	3,84	3,97
170	3,76	3,52	3,55	3,62	3,83	3,97
180	3,76	3,52	3,55	3,63	3,83	3,97
190	3,75	3,52	3,55	3,63	3,84	3,97
200	3,76	3,51	3,55	3,62	3,83	3,97

Tabla A.22: Número de CAs vs. mensajes firmados emitidos por cada CA

CAs	MENSAJES FIRMADOS POR CA					
	rel=0,1	**rel=0,3**	**rel=0,4**	**rel=0,5**	**rel=0,7**	**rel=0,9**
10	4,31	5,38	5,45	5,43	5,47	5,68
20	5,30	5,46	5,51	5,55	5,61	5,84
30	5,45	5,49	5,46	5,58	5,67	5,87
40	5,45	5,45	5,50	5,56	5,71	5, 90
50	5,45	5,46	5,52	5,54	5,65	5,90
60	5,46	5,46	5,49	5,57	5,66	5,89
70	5,40	5,43	5,47	5,48	5,67	5,92
80	5,43	5,42	5,47	5,55	5,65	5,87
90	5,44	5,41	5,43	5,52	5,64	5,91
100	5,45	5,43	5,46	5,49	5,64	5,89
110	5,43	5,43	5,43	5,48	5,70	5,89
120	5,43	5,42	5,47	5,48	5,71	5,89
130	5,42	5,42	5,47	5,50	5,68	5,90
140	5,44	5,44	5,39	5,48	5,63	5,92
150	5,45	5,45	5,46	5,47	5,64	5,90
160	5,38	5,44	5,47	5,52	5,60	5,90
170	5,43	5,39	5,44	5,51	5,652	5,88
180	5,41	5,41	5,49	5,45	5,67	5,90
190	5,44	5,36	5,41	5,52	5,64	5,90
200	5,45	5,42	5,39	5,51	5,63	5,92

Tabla A.23: Número de CAs vs. número de raíces cuando varía L_{MAX}

CAs	NÚMERO DE RAÍCES					
	L_{MAX}=2	L_{MAX}=3	L_{MAX}=4	L_{MAX}=5	L_{MAX}=6	L_{MAX}=7
10	1,32	1,14	1,12	1,12	1,10	1,02
20	1,36	1,12	1,38	1,28	1,24	1,10
30	1,66	1,32	1,22	1,30	1,26	1,26
40	1,78	1,36	1,28	1,24	1,30	1,32
50	1,74	1,28	1,34	1,32	1,20	1,40
60	2,08	1,52	1,52	1,42	1,28	1,32
70	2,00	1,40	1,36	1,44	1,32	1,32
80	2,12	1,50	1,44	1,32	1,36	1,44
90	1,96	1,52	1,38	1,36	1,56	1,38
100	2,20	1,50	1,42	1,36	1,44	1,28
110	2,16	1,50	1,34	1,38	1,34	1,54
120	2,14	1,54	1,44	1,46	1,26	1,32
130	2,34	1,58	1,58	1,56	1,34	1,46
140	2,30	1,58	1,32	1,38	1,50	1,38
150	2,24	1,46	1,46	1,42	1,44	1,40
160	2,38	1,54	1,62	1,5	1,6	1,54
170	2,52	1,54	1,46	1,54	1,42	1,46
180	2,42	1,56	1,46	1,48	1,48	1,4
190	2,40	1,64	1,48	1,50	1,36	1,38
200	2,40	1,66	1,48	1,46	1,50	1,42

Tabla A.24: Número de CAs vs. probabilidad de fallo cuando varía L_{MAX}

CAs	PROBABILIDAD DE FALLO					
	L_{MAX}=2	L_{MAX}=3	L_{MAX}=4	L_{MAX}=5	L_{MAX}=6	L_{MAX}=7
10	0,10	0,05	0,04	0,04	0,03	0,00
20	0,13	0,03	0,10	0,08	0,09	0,03
30	0,20	0,07	0,04	0,08	0,07	0,06
40	0,24	0,12	0,09	0,06	0,10	0,10
50	0,24	0,09	0,11	0,10	0,06	0,12
60	0,33	0,16	0,12	0,14	0,08	0,10
70	0,29	0,12	0,10	0,13	0,08	0,09
80	0,33	0,13	0,11	0,08	0,11	0,09
90	0,25	0,15	0,11	0,09	0,14	0,11
100	0,29	0,12	0,12	0,13	0,12	0,09
110	0,32	0,12	0,09	0,13	0,11	0,16
120	0,31	0,13	0,12	0,13	0,08	0,09
130	0,33	0,18	0,16	0,13	0,09	0,14
140	0,32	0,16	0,11	0,12	0,13	0,11
150	0,37	0,13	0,14	0,12	0,13	0,14
160	0,38	0,16	0,16	0,12	0,16	0,15
170	0,33	0,17	0,14	0,19	0,13	0,13
180	0,39	0,19	0,14	0,15	0,16	0,09
190	0,32	0,20	0,15	0,15	0,09	0,13
200	0,35	0,16	0,11	0,12	0,15	0,12

Tabla A.25: Número de CAs vs. longitud promedio de caminos cuando varía L_{MAX}

CAs	LONGITUD PROMEDIO DE CAMINOS					
	L_{MAX}=2	L_{MAX}=3	L_{MAX}=4	L_{MAX}=5	L_{MAX}=6	L_{MAX}=7
10	1,29	1,42	1,40	1,35	1,43	1,42
20	1,45	1,55	1,52	1,52	1,50	1,58
30	1,52	1,66	1,62	1,52	1,68	1,56
40	1,52	1,56	1,53	1,63	1,57	1,54
50	1,56	1,64	1,56	1,62	1,71	1,61
60	1,54	1,58	1,61	1,63	1,66	1,62
70	1,53	1,62	1,64	1,70	1,64	1,64
80	1,56	1,61	1,64	1,71	1,66	1,71
90	1,68	1,65	1,70	1,72	1,64	1,66
100	1,63	1,71	1,63	1,65	1,75	1,73
110	1,67	1,70	1,70	1,62	1,72	1,62
120	1,66	1,65	1,65	1,64	1,78	1,74
130	1,61	1,56	1,65	1,66	1,77	1,73
140	1,67	1,64	1,77	1,73	1,69	1,71
150	1,68	1,72	1,65	1,75	1,64	1,72
160	1,63	1,66	1,65	1,70	1,70	1,67
170	1,66	1,71	1,68	1,64	1,77	1,71
180	1,71	1,68	1,60	1,75	1,61	1,71
190	1,65	1,65	1,74	1,73	1,80	1,71
200	1,62	1,71	1,80	1,66	1,73	1,74

Tabla A.26: Número de CAs vs. número de rondas cuando varía L_{MAX}

CAs	NÚMERO DE RONDAS					
	L_{MAX}=2	L_{MAX}=3	L_{MAX}=4	L_{MAX}=5	L_{MAX}=6	L_{MAX}=7
10	15,46	14,46	14,22	13,98	14,44	14,00
20	23,90	22,76	22,22	22,06	22,00	21,60
30	31,84	30,90	29,92	29,14	30,02	29,16
40	39,08	37,44	35,44	36,54	36,12	36,44
50	47,12	43,58	43,38	42,94	44,02	43,34
60	54,12	51,26	50,32	50,50	51,14	50,42
70	60,80	57,88	57,24	57,58	58,36	57,36
80	67,60	65,48	64,10	64,84	64,46	64,88
90	74,18	72,82	72,32	72,64	71,28	71,70
100	82,04	80,16	78,86	78,06	79,52	79,78
110	90,08	86,62	86,50	85,34	85,26	85,36
120	95,98	94,94	92,36	92,64	92,36	92,02
130	103,06	100,98	100,68	98,90	99,20	100,10
140	110,64	108,28	106,38	107,38	106,06	106,46
150	117,14	115,26	113,22	113,84	113,14	113,78
160	125,22	121,00	118,64	121,32	120,34	119,90
170	131,64	128,94	126,54	127,50	127,56	126,64
180	138,30	136,64	132,62	134,16	133,30	132,88
190	146,16	142,48	141,72	141,80	142,92	141,70
200	153,14	148,08	149,18	147,72	149,24	149,02

Tabla A.27: Número de CAs vs. número de mensajes enviados por CA cuando varía L_{MAX}

CAs	MENSAJES ENVIADOS POR CA					
	L_{MAX}=2	L_{MAX}=3	L_{MAX}=4	L_{MAX}=5	L_{MAX}=6	L_{MAX}=7
10	23,43	22,35	22,11	21,97	22,22	22,21
20	46,86	45,70	45,44	45,04	45,05	45,13
30	70,86	69,25	68,64	67,90	68,58	67,70
40	93,88	91,14	90,61	90,40	90,61	90,32
50	117,01	113,65	113,44	113,30	113,20	113,29
60	139,96	136,88	136,21	136,18	136,01	135,62
70	162,10	159,53	158,60	159,30	158,44	158,50
80	185,19	181,89	181,74	181,40	181,63	181,57
90	208,18	204,70	204,26	204,59	203,86	203,99
100	230,57	227,97	226,69	225,85	227,47	226,97
110	253,61	250,22	249,64	248,41	249,31	248,99
120	276,60	273,33	271,90	271,85	272,28	272,19
130	298,92	295,37	295,38	294,47	295,27	295,17
140	322,47	318,24	317,38	317,46	318,01	317,42
150	344,52	341,63	340,43	339,81	340,10	340,24
160	367,33	363,83	363,09	362,58	363,31	363,23
170	390,48	386,58	385,54	385,57	386,12	385,01
180	413,37	409,46	408,15	407,40	407,56	408,52
190	435,67	431,73	430,62	431,12	430,86	430,23
200	458,86	455,06	454,00	453,38	453,22	453,23

Tabla A.28: Número de CAs vs. número de mensajes firmados por CA cuando varía L_{MAX}

CAs	MENSAJES FIRMADOS POR CA					
	L_{MAX}=2	L_{MAX}=3	L_{MAX}=4	L_{MAX}=5	L_{MAX}=6	L_{MAX}=7
10	5,80	5,44	5,45	5,39	5,40	5,36
20	5,73	5,50	5,51	5,49	5,50	5,46
30	5,68	5,47	5,46	5,51	5,43	5,49
40	5,65	5,52	5,50	5,46	5,49	5,51
50	5,59	5,46	5,52	5,47	5,41	5,48
60	5,59	5,50	5,49	5,48	5,44	5,48
70	5,60	5,47	5,47	5,42	5,47	5,47
80	5,56	5,50	5,47	5,42	5,45	5,43
90	5,43	5,46	5,43	5,42	5,47	5,46
100	5,48	5,43	5,46	5,46	5,41	5,42
110	5,43	5,41	5,43	5,49	5,42	5,49
120	5,44	5,44	5,47	5,48	5,38	5,40
130	5,48	5,52	5,47	5,46	5,40	5,43
140	5,42	5,45	5,39	5,42	5,45	5,43
150	5,40	5,40	5,46	5,42	5,47	5,43
160	5,45	5,43	5,47	5,45	5,43	5,44
170	5,42	5,40	5,44	5,49	5,39	5,44
180	5,37	5,43	5,49	5,40	5,49	5,43
190	5,41	5,43	5,41	5,43	5,37	5,43
200	5,45	5,40	5,39	5,45	5,43	5,41

Apéndice B

Fases de PROSEARCH y Algoritmos

Este apéndice contiene los algoritmos que describen el funcionamiento de cada fase de PROSEARCH desde el punto de vista de una autoridad.

La Tabla B.1 contiene la notación utilizada en estos algoritmos:

Tabla B.1: Notación algoritmos

Notación	Significado
L_{MAX}	Longitud de camino máxima permitida
L_M	Longitud máxima de camino permitida en la segunda fase
CA_i	Autoridad de certificación i
L_i	Número de certificados de las hojas a la autoridad i
IN_i	Número de autoridades en que CA_i confía (certificados recibidos)
OUT_i	Número de autoridades que confían en CA_i (certificados expedidos)
CA_0	Autoridad actual
N_0	Número de vecinos participantes de CA_0
Order[N_0+1]	Arreglo que contiene CA_0 y sus vecinos participantes ordenados desde el menos confiable al más confiable.
pos	Posición de CA_0 dentro del arreglo Order

B.1 Orden de Confiabilidad entre las CAs

El siguiente algoritmo describe el procedimiento llevado a cabo por cada autoridad en la primera fase de PROSEARCH.

```
BEGIN
    IF CA0 begins the protocol THEN
      CA0 CHOOSE LMAX
      CA0 SEND request message TO all its neighbors
      CA0 RECEIVE acceptance/rejection messages FROM its
      neighbors
    ELSE
```

```
      CA0 RECEIVE request message FROM some neighbor
      IF CA0 does not want to be part of the hierarchy
      THEN
        CA0 SEND rejection message TO its demanding
        neighbor
        CA0 finishes the protocol
        GO TO END
      ELSE
        CA0 SEND acceptance message TO its demanding
        neighbor
        CA0 SEND request message TO its other neighbors
        CA0 RECEIVE acceptance/rejection messages FROM
        its neighbors
      END IF
    END IF
    CA0 COMPUTE IN0 and OUT0
    CA0 SEND IN0, OUT0 TO its participant neighbors
    CA0 RECEIVE INi, OUTi FROM its participant neighbors
    Order[1]=0 /*CA0 in position 1 of Order array*/
    pos=1
    FOR j=1 TO N0
      IF OUT0 < OUTj THEN
        GO TO POST
      ELSE IF OUT0 = OUTj THEN
        IF IN0 <= INj THEN
POST:     CA0 is less trustworthy than CAj
          FOR p=N0+1 TO pos+2
            Order[p]=Order[p-1]
          END FOR
          Order[pos+1] = j
          FOR k=(pos+1) TO N0
            p1=Order[k]
            p2=Order[k+1]
            IF OUTp1 > OUTp2 THEN
              p1 is more trustworthy than p2
              Order[k]=p2
              Order[k+1]=p1
            ELSE IF OUTp1 = OUTp2 THEN
              IF INp1 > INp2 THEN
                p1 is more trustworthy than p2
                Order[k]=p2
                Order[k+1]=p1
              END IF
            END IF
          END FOR
        ELSE
          GO TO PREV
        END IF
      ELSE IF OUT0 > OUTj THEN
PREV:   CA0 is more trustworthy than CAj
```

```
          pos = pos+1
          FOR p=N0+1 TO pos
            Order[p]=Order[p-1]
          END FOR
          Order[pos-1] = j
          IF (pos-1)>1 THEN
            FOR k=(pos-1) TO 2
              p1=Order[k]
              p2=Order[k-1]
              IF OUTp1 < OUTp2 THEN
                p1 is less trustworthy than p2
                Order[k]=p2
                Order[k-1]=p1
              ELSE IF OUTp1 = OUTp2 THEN
                IF INp1 <= INp2 THEN
                  p1 is less trustworthy than p2
                  Order[k]=p2
                  Order[k-1]=p1
                END IF
              END IF
            END FOR
          END IF
        END IF
      END FOR
END
```

B.2 Creación de la Jerarquía

El siguiente algoritmo describe el procedimiento llevado a cabo por cada autoridad en la segunda fase de PROSEARCH.

```
BEGIN
    IF protocol repetition THEN
      LM = LMAX
    ELSE
      LM=LMAX-1
    END IF
    IF CA0 is the most trustworthy neighbor THEN
      CA0 is root CA
      CA0 SEND root_CA message TO its neighbors
      IF CA0 RECEIVED failure message THEN
        root CAs REPEAT the protocol
      ELSE
        CA0 is the root of the hierarchy
        CA0 SEND root_CERT message TO all its subordinate
        authorities
```

```
      END IF
    ELSE IF CA0 is the less trustworthy neighbor THEN
      GO TO ACT
    ELSE
      CA0 RECEIVE association message FROM its less
      trustworthy neighbors
ACT:  FOR j = N0+1 TO pos+1
        t=Order[j]
        IF CA0 trust CAt THEN
          n = Lt
          IF Lt <= L0 THEN
            Lt=L0+1
          END IF
          IF Lt <= LM THEN
            CA0 CHOOSE CAt like Superior CA
            BREAK /*exit FOR*/
          ELSE
            Lt = n
            CA0 must choose another superior CA
          END IF
        END IF
      END FOR
      IF CA0 did not choose a superior CA THEN
        CA0 SEND failure message TO its neighbors
        CA0 is root CA
      ELSE
        CA0 SEND association message TO all its neighbors
      END IF
    END IF
END
```

Bibliografía

[ACG04] G. Anastasi, M. Conti and E. Gregori, *Mobile Ad Hoc Networking - Chapter 3: IEEE 802.11 AD HOC Networks: Protocols, Performance, and Open Issues*, Wiley-Interscience, 2004.

[AL03] C. Adams and S. Lloyd, *Understanding PKI: Concepts, Standards, and Deployment Considerations*, 2nd Ed., Addison-Wesley, 2003.

[ANS99] ANSI,"Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", 1999.

[Aur97] T. Aura, "Comparison of Graph-Search Algorithms for Authorization Verification in Delegation Networks", *2nd Nordic Workshop on Secure Computer Systems (NORDSEC'97)*, Espoo(Finland), 1997.

[AVT04] P. G. Argyroudis, R. Verma, H. Tewari and D. O'Mahony, "Performance Analysis of Cryptographic Protocols on Handheld Devices", *Third IEEE International Symposium on Network Computing and Applications (NCA'04)*, pp. 169-174, 2004.

[BG04] C. Budakoglu and T. A. Gulliver, "Hierarchical key management for mobile ad-hoc networks", *2004 IEEE 60th Vehicular Technology Conference (VTC2004-Fall)*, pp. 2735-2738 Vol.4, 2004.

[BHR99] S. Boeyen, T. Howes and P. Richard,"RFC2587 - Internet X.509 Public Key Infrastructure LDAPv2 Schema", *IETF Network Working Group*, 1999.

[BK03] J. D. Beatty and J. Kemp,"Liberty Protocols and Schema Specification Version 1.1", *Liberty Alliance Project*, 2003.

[CBV04] P. Chatzimisios, A. C. Boucouvalas and V. Vitsas, "Optimisation of RTS/CTS Handshake in IEEE 802.11Wireless LANs for Maximum Performance", *IEEE*

Global Telecommunications Conference Workshops, 2004 (GlobeCom Workshops 2004), pp. 270-275, 2004.

[Cer00] Certicom,"Standards for Efficient Cryptography SEC 1: Elliptic Curve Cryptography Version 1.0", 2000, http://www.secg.org.

[CG03] O. Cánovas and A. F. G. Skarmeta, "Retos y Oportunidades de los Sistemas de Control de Acceso basados en Delegación", *Jornadas Técnicas RedIRIS*, 2003.

[CL01] R. Castro-Rojo and D. R. López, "The PAPI System: Point of Access to Providers of Information", *The International Journal of Computer and Telecommunications Networking*, vol. 37(6), pp. 703-710, 2001.

[Cle02] J. D. Clercq, "Single Sign-On Architectures", *InfraSec 2002*, Springer -Verlag, LNCS 2437, pp. 40-58, 2002.

[Com06] *Compras y Tiendas Virtuales: ¿Qué es una Tienda Virtual?*, Fecha de acceso: 29/08/2006, http://www.comercio-electronico-centro-comercial.com/tienda/compra-venta/informatica-ocio/tienda_virtual.htm.

[Coo99] D. A. Cooper, "A Model of Certificate Revocation", *15th Annual Computer Security Applications Conference(ACSAC '99)*, pp. 256-264, 1999.

[DA99] T. Dierks and C. Allen,"RFC2246 - The TLS Protocol Version 1.0", *IETF Network Working Group*, 1999.

[Dai04] W. Dai, *Crypto ++ 5.2.1 Benchmarks*, Fecha de acceso: 4/10/2004, http://www.eskimo.com/~weidai/benchmarks.

[DGS05] Y. Dong, H. W. Go, A. F. Sui, V. O. K. Li, L. C. K. Hui and S. M. Yiu, "Providing Distributed Certificate Authority Service in Mobile Ad Hoc Networks", *First International Conference on Security and Privacy for Emerging Areas in Communications Networks 2005 (SecureComm 2005)*, pp. 149-156, 2005.

[DH76] W. Diffie and M. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, vol. 22, pp. 644-654, 1976.

[DRW04] D. Dhillon, T. S. Randhawa, M. Wang and L. Lamont, "Implementing a Fully Distributed Certificate Authority in an OLSR MANET", *2004 IEEE Wireless Communications and Networking Conference 2004 (WCNC)*, pp. 682-688 Vol. 2, 2004.

[EAH01] Y. Elley, A. Anderson, S. Hanna, S. Mullan, R. Perlman and S. Proctor, "Building Certification Paths: Forward vs. Reverse", *Network and Distributed System Security Symposium (NDSS 2001)*, 2001.

[EC05] L. Ertaul and N. Chavan, "Security of Ad-Hoc Networks and Threshold Cryptography", *International Conference on Wireless Networks, Communications and Mobile Computing (WIRELESSCOM 2005)*, vol. 1, pp. 69-74, 2005.

[EFL99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas and T. Ylonen,"RFC2693 - SPKI Certificate Theory", *IETF Network Working Group*, 1999.

[Ell99] C. Ellison,"RFC2692 - SPKI Requirements", *IETF Network Working Group*, 1999.

[GGC02] V. Gupta, S. Gupta, S. Chang and D. Stebila, "Performance Analysis of Elliptic Curve Cryptography for SSL", *3rd ACM Workshop on Wireless Security*, pp. 87-94, 2002.

[Gig05] J. Gigli, *España: Safelayer Suministrará la PKI que Sustentará el Futuro Documento Nacional de Identidad*, Fecha de acceso: 30/08/2006, http://www.gobiernoelectronico.org/?q=node/3257.

[HBC01] J.-P. Hubaux, L. Buttyán and S. Capkun, "The Quest for Security in Mobile Ad-Hoc Networks", *ACM Symposium on Mobile Ad-Hoc Networking and Computing (MobiHOC 2001)*, 2001.

[HC98] D. Harkins and D. Carrel,"RFC2409 - The Internet Key Exchange (IKE)", *IETF Network Working Group*, 1998.

[HM02] P. Hallam-Baker and E. Maler,"Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)", *OASIS*, 2002.

[HPF02] R. Housley, W. Polk, W. Ford and D. Solo,"RFC3280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", *IETF Network Working Group*, 2002.

[HPF99] R. Housley, W. Polk, W. Ford and D. Solo,"RFC 2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile", *IETF Network Working Group*, 1999.

[HPS05] J. Hernandez-Serrano, J. Pegueroles and M. Soriano, "Algoritmo para la Creación de una Infraestructura Fija Virtual para Gestión de Claves en Grandes Grupos sobre MANET", *V Jornadas de Ingeniería Telemática (JITEL 2005)*, Universidad de Vigo, pp. 173-180, Vigo(España), 2005.

[Hun02] B. Hunter, "Simplifying PKI Usage Through a Client-Server Architecture and Dynamic Propagation of Certificate Paths and Repository Addresses", *13th International Workshop on Database and Expert System Applications (DEXA'02)*, pp. 425-430, 2002.

[ITU00] ITU-T,"Recommendation X.509: Information Processing Systems - Open Systems Interconnection - The Directory : Authentication Framework (Technical Corrigendum)", *International Telecommunication Union*, 2000.

[ITU01] ITU-T,"Recommendation X.500: Information Technology -Open Systems Interconnection - The Directory - Overview of Concepts, Models, and Services", *International Telecommunication Union*, Geneva (Switzerland), 2001.

[KAG98] G. Karjoth, N. Asokan and C. Gülcü, "Protecting the Computation Results of Free-Roaming Agents", *Second International Workshop on Mobile Agents (MA'98)*, Springer-Verlag, LNCS1477, pp. 195-207, London(UK), 1998.

[Ken93] S. Kent,"RFC 1422 - Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management", *IETF Network Working Group*, 1993.

[Kil95] S. Kille,"RFC 1779 - A String Representation of Distinguished Names", *IETF Network Working Group*, 1995.

[KLL02] S. Kiran, P. Lareau and S. Lloyd,"PKI Basics - A Technical Perspective", *PKI Forum*, 2002, http://www.pkiforum.org.

[Koc98] P. C. Kocher, "On Certificate Revocation and Validation", *Second International Conference on Financial Cryptography*, LNCS, vol. 1465, pp. 172 - 177, Springer-Verlag, 1998.

[Koh78] L. M. Kohnfelder,"Towards a Practical Public-Key Cryptosystem", MIT Laboratory for Computer Science, 1978.

[KZL01] J. Kong, P. Zerfos, H. Luo, S. Lu and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks", *Ninth International Conference in Network Protocols (ICNP'01)*, pp. 251-260, 2001.

[Lam71] B. Lampson, "Protection", *5th Princeton Symposium on Information Science and Systems.*, Reprinted in ACM Operating Systems Review, pp. 437-443, 1971.

[Lam81] L. Lamport, "Password Authentication with Insecure Communication", *Communications of the ACM*, vol. 24, pp. 770-772, 1981.

[LC00] A. Levi and M. U. Caglayan, "An Efficient, Dynamic and Trust Preserving Public Key Infrastructure", *2000 IEEE Symposium on Security and Privacy (S&P 2000)*, pp. 203-214, 2000.

[LC99] A. Levi and M. U. Caglayan, "Verification of Classical Certificates Via Nested Certificates and Nested Certificate Paths", *Eight International Conference on Computer Communications and Networks*, pp. 242-247, 1999.

[Lin00] J. Linn,"Trust Models and Management in Public-Key Infrastructures", *RSA Laboratories*, Technical Note, 2000. http://www.rsasecurity.com/rsalabs/, http://www.rsasecurity.com/rsalabs/.

[Luc99] M. J. Lucena, *Criptografía y Seguridad en Computadores*, 2da Ed., Universidad de Jaén, Jaén, 1999.

[MAM99] M. Myers, R. Ankney, A. Malpani, S. Galperin and C. Adams,"RFC2560 - X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", *IETF Network Working Group*, 1999.

[MC04] D. Mundy and D. W. Chadwick, "An XML Alternative for Performance and Security: ASN.1", *IT Professional*, vol. 6(1), pp. 30-36, 2004.

[Mer89] R. C. Merkle, "A Certified Digital Signature", *9th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO'89)*, Lecture Notes in Computer Science, vol. 435, pp. 218 - 238, Springer-Verlag, 1989.

[MF02] J. L. Muñoz and J. Forné, "Evaluation of Certificate Revocation Policies: OCSP vs. Overissued CRL", *Workshop on Trust and Privacy in Digital Business (TrustBus02)*, IEEE Computer Society, pp. 511-515, 2002.

[MFE03] 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 (IJIS)*, 2003.

[Mic05] S. Micali,"Enhanced Certificate Revocation System", *Technical memo MIT/LCS/TM-542*, 1995.

[MM02] E. Martín and F. Marcelo, "PKI y Certificados Digitales: Un Mercado en Alza", *Comunicaciones World*, vol. 167, 2002.

[MOV97] A. J. Menezes, P. C. v. Oorschot and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1997.

[MS02] J. Marchesini and S. Smith, "Virtual Hierarchies - An Architecture for Building and Maintaining Efficient and Resilient Trust Chains", *7th Nordic Workshop on Secure IT Systems (NORDSEC 2002)*, Karlstad (Sweden), 2002.

[NCS88] NCSC, *Glossary of Computer Security Terms*, Fecha de acceso: 24/02/2006, http://csrc.nist.gov/secpubs/rainbow/tg004.txt.

[NIS00] NIST,"Digital Signature Standard (DSS)", FIPS PUB 186, 2000.

[NIS01] NIST,"Advanced Encryption Standard(AES)", FIPS PUB 46-1, 2001.

[NIS88] NIST,"Data Encryption Standard(DES)", FIPS PUB 46-1, 1988.

[NIS95] NIST,"Secure Hash Standard", FIPS PUB 180-1, 1995.

[NIS99] NIST,"Triple DES", FIPS PUB 46-3, 1999.

[NN98] M. Naor and K. Nissim, "Certificate Revocation and Certificate Update", *7th USENIX Security Symposium*, pp. 217-228, 1998.

[Per99] R. Perlman, "An Overview of PKI Trust Models", *IEEE Network*, vol. 13, pp. 38-43, 1999.

[PH00] W. T. Polk and N. E. Hastings,"Bridge Certification Authorities: Connecting B2B Public Key Infrastructures", *NIST*, 2000.

[PH02] D. Pinkas and R. Housley,"RFC 3379 - Delegated Path Validation and Delegated Path Discovery Protocol Requirements", *IETF Network Working Group*, 2002.

[PLZ05] H. Pan, J. Li, Y. Zhu and D. Wei, "A Practical Scheme of Merging Multiple Public Key Infrastructure in E-commerce", *Networking and Mobile Computing: 3rd International Conference (ICCNMC 2005)*, Springer-Verlag, LNCS 3619, pp. 1287-1294, Zhangjiajie(China), 2005.

[Riv92] R. Rivest,"RFC 1321 - The MD5 Message-Digest Algorithm", *IETF Working Group*, 1992.

[RL96] R. L. Rivest and B. Lampson, *SDSI- A Simple Distributed Security Infrastructure*, Fecha de acceso: 28/11/2006, http://theory.lcs.mit.edu/~rivest/sdsi10.html.

[RSA78] R. L. Rivest, A. Shamir and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Communications of the ACM*, vol. 21(2), pp. 120-126, 1978.

[Saf05] Safelayer, *Safelayer Reforzará su Liderazgo en Tecnología PKI, con Calidad y Sencillez*, Fecha de acceso: 30/08/2006, http://www.safelayer.com/content/view/68/55/lang,es/.

[San93] R. S. Sandhu, "Lattice-based access control models", *Computer*, vol. 26, pp. 9-19, 1993.

[SC05] T. Scavo and S. Cantor,"Shibboleth Architecture. Technical Overview", *Working Draft 02, 8 June 2005*, http://shibboleth.internet2.edu/docs/draft-mace-shibboleth-tech-overview-latest.pdf.

[SF04] C. Satizábal and J. Forné, "Revocación de Certificados en la Validación de Caminos de Certificación", *VIII Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2004)*, Ediciones Díaz de Santos S.A., vol. 1, Avances en Criptología y Seguridad de la Información, pp. 605-613, Madrid (España), 2004.

[SFH06] C. Satizábal, J. Forné, J. Hernández-Serrano and J. Pegueroles, "Building Hierarchical Public Key Infrastructures in Mobile Ad-Hoc Networks", *The 2nd International Conference on Mobile Ad-Hoc and Sensor Networks (MSN 2006)*, LNCS, vol. 4325, pp. 485-496, Hong Kong (China), 2006.

[Sha79] A. Shamir,"How to Share a Secret", Communications of the ACM, vol. 22, pp. 612-613, 1979.

[SHF] C. Satizábal, J. Hernández-Serrano, J. Forné and J. Pegueroles, "Building a Virtual Hierarchy to Simplify Certification Path Discovery in Mobile Ad-Hoc Networks", *Computer Communications*.

[Shi00] R. Shirey,"RFC2828 - Internet Security Glossary", *IETF Network Working Group*, 2000.

[SPF05a] C. Satizábal, R. Páez and J. Forné, "Privilege Delegation: Path Validation vs. Certificate Revocation", *IV Congreso Internacional de Electrónica y Tecnologías de Avanzada*, Revista Colombiana de Tecnologías de Avanzada, vol. 1, pp. 8-14, Pamplona (Colombia), 2005.

[SPF05b] C. Satizábal, R. Páez and J. Forné, "Certification Path Validation in WPKI", *Infocom 2005 Student Workshop*, Miami (USA), 2005.

[SPF05c] C. Satizábal, R. Páez and J. Forné, "PKI Trust Relationship Using Hash Chains", *International Conference on Advances in the Internet, Processing, Systems and Interdisciplinary Research (IPSI 2005)*, Carcassonne (Francia), 2005.

[SPF05d] C. Satizábal, R. Páez and J. Forné, "WAP PKI and Certification Path Validation", *11th Open European Summer School : Networked Applications (EUNICE 2005)*, pp. 185-190, Colmenarejo (España), 2005.

[SPF05e] C. Satizábal, R. Páez and J. Forné, "Construyendo Caminos de Certificación Mediante Cadenas de Hash", *V Jornadas de Ingeniería Telemática (JITEL 2005)*, pp. 343-350, Vigo (España), 2005.

[SPF06a] C. Satizábal, R. Páez and J. Forné, "PKI Trust Relationships: From a Hybrid Architecture to a Hierarchical Model", *The First International Conference on Availability, Reliability and Security (ARES 2006)*, IEEE Computer Society, pp. 563-570, Vienna University of Technology (Austria), 2006.

[SPF06b] C. Satizábal, R. Páez and J. Forné, "PROSEARCH: A Protocol to Simplify Path Discovery in Critical Scenarios", *1st International Workshop on Critical Information Infrastructures Security (CRITIS'06)*, LNCS, vol. 4347, pp. 151-165, Samos Island (Grecia), 2006.

[SPF06c] C. Satizábal, R. Páez and J. Forné, "Building a Virtual Hierarchy for Managing Trust Relationships in a Hybrid Architecture", *Journal of Computers (JCP)*, vol. 1(7), pp. 60-68, 2006.

[SPF07] C. Satizábal, R. Páez and J. Forné, "WAP PKI and Certification Path Validation", *International Journal of Internet Protocol Technology (IJIPT)*, vol. 1(2), 2007.

[SSW05] A. J. Stell, R. O. Sinnott and J. P. Watt, "Comparison of Advanced Authorisation Infrastructures for Grid Computing", *19th International Symposium on High Performing Computing Systems and Applications (HPCS'05)*, pp. 195-201, 2005.

[TG04] S. Tillich and J. Grobschädl, "A Survey of Public-Key Cryptography on J2ME-Enabled Mobile Devices", *19th International Symposium on Computer an Information Sciences (ISCIS 2004)*, Springer-Verlag Heidelberg, LNCS 3280, pp. 935-944, Kemer-Antalya(Turkey), 2004.

[Van03] S. A. Vanstone, "Next Generation Security for Wireless: Elliptic Curve Cryptography", *Computers & Security*, vol. 22(5), pp. 412-415, 2003.

[Ver03] Verisign,"VeriSign Trust Network Certificate Policies Version 1.2", *Verisign Inc.*, 2003, http://www.verisign.com/.

[Ver05] Verisign, *Verisign Trust Network(VTN) Key Hierarchy*, Fecha de acceso: 21/10/2006, http://www.verisign.com/repository/hierarchy/hierarchy.pdf.

[VSJ05] S. Vimercati, P. Samarati and S. Jajodia, "Policies, Models, and Languages for Access Control", *4th International Workshop on Databases in Networked Information Systems (DNIS 2005)*, LNCS, vol. 3433, pp. 225-237, 2005.

[WAP01a] WAPForum,"Wireless Application Protocol Architecture Specification", *WAPForum*, Specification WAP-210-WAPArch-20010712, 2001, http://www.wapforum.org/.

[WAP01b] WAPForum,"Wireless Application Protocol Public Key Infrastructure Definition", *WAPForum*, Specification WAP-217-WPKI-20010424-a, 2001, http://www.wapforum.org/.

[WAP01c] WAPForum,"Wireless Markup Language Version 2.0", *WAPForum*, Specification WAP-238-WML-20010911-a, 2001, http://www.wapforum.org/.

[WAP01d] WAPForum,"WMLScript Crypto Library", *WAPForum*, Specification WAP-161-WMLScriptCrypto-20010620-a, 2001, http://www.wapforum.org/,.

[WAP01e] WAPForum,"Wireless Transport Layer Security", *WAPForum*, Specification WAP-261-WTLS-20010406-a, 2001, http://www.wapforum.org/.

[WAP01f] WAPForum,"Wireless Identity Module Part: Security", Specification WAP-260-WIM-20010712-a, 2001, http://www.wapforum.org/.

[WAP01g] WAPForum,"WAP Certificate and CRL Profiles", Specification WAP-211-WAPCert-20010522-a, 2001, http://www.wapforum.org/.

[WHK97] M. Wahl, T. Howes and S. Kille,"RFC2251: Lightweight Directory Access Protocol (v3)", *IETF Network Working Group*, 1997.

[YHK95] W. Yeong, T. Howes and S. Kille,"Lightweight Directory Access Protocol", *Performance Systems International, University of Michigan and ISODE Consortium*, 1995.

[YHK95] W. Yeong, T. Howes and S. Kille,"RFC1777: Lightweight Directory Access Protocol", *IETF Network Working Group*, 1995.

[YK03] S. Yi and R. Kravets, "MOCA: Mobile Certificate Authority for Wireless Ad-hoc Networks", *Communications of the ACM*, 2003.

[YT05] E. Yuan and J. Tong, "Attributed Based Access Control (ABAC) for Web Services", *IEEE International Conference on Web Services (ICWS'05)*, pp. 569-577, 2005.

[Zim05] P. R. Zimmermann, *The Official PGP User's Guide*, MIT Press, 1995.