Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

IMPLEMENTACIÓN DEL PROTOCOLO SECURE NEIGHBOR DISCOVERY (SEND)

Trabajo Especial de Grado
presentado ante la ilustre
Universidad Central de Venezuela
por el Bachiller

Say Hood. T. Chiu A.
C.I.: 16.804.067
e-mail: saychiu@gmail.com

para optar al título de Licenciado en Computación

Tutor: Eric Gamess

Caracas, Noviembre 2008

Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Laboratorio de Comunicación y Redes

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designados por el Consejo de Escuela de Computación, para examinar el Trabajo Especial de Grado, presentado por el Bachiller Say Hood Chiu Acosta C.I. 16.804.067, con el título "**Implementación del Protocolo Neighbor Discovery (SEND)**", a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 14 de noviembre de 2008, a las 3:00pm, para que su autor lo defendiera en forma pública, en el Laboratorio de Internet2, lo cual este realizó mediante una exposición oral de su contenido, y luego respondió satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela. Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente acta, en Caracas el 14 de noviembre de 2008, dejándose también constancia de que actuó como Coordinador del Jurado el Profesor Tutor Eric Gamess.

Prof. Eric Gamess
(Tutor)

Prof. David Pérez
(Jurado Principal)

Prof. Karima Velásquez
(Jurado Principal)

Dedicatoria

A mis padres Wing y Evelia, quienes me ayudaron con su esfuerzo, apoyo, confianza, dedicación y amor incondicional a lograr mis metas.

Este trabajo especial de grado es para ustedes.

Los AMO...

Agradecimientos

A Dios, por haberme dado la fuerza, paciencia, constancia y perseverancia necesaria para alcanzar mis metas. Siempre confiaré en ti Señor...

A mis hermanos Wing, Waid y Meylin, por esa confianza que tienen en mi, por el apoyo incondicional brindado y sobre todo por quererme como lo hacen.

A mi madrina Nely, quién me apoyó y orientó desde mi comienzo en la universidad, abriéndome las puertas de su casa.

A mi abuelita Celia, por haberme acogido como un hijo más en su casa, sirviéndome con todo su amor y cariño.

A tía Zulay, quien aguantó todas mis bromas y malos ratos durante toda la carrera.

A tía Argelia y Vicente, por estar pendientes de mi y haberme ayudado cuando los necesite.

A mi profesor, amigo y tutor Eric Gamess por haberme ayudado desde mis comienzos en la DTIC, gracias por su paciencia y conocimientos ofrecidos.

A mis tíos, tías, primos y amigos, por brindarme su ayuda y apoyo cuando los necesité.

Y por último a todos aquellos que de una u otra manera me ayudaron y apoyaron a lo largo de mi carrera.

Gracias a todos por ser parte importante en mi vida...

Resumen

Titulo:
Implementación del Protocolo Secure Neighbor Discovery (SEND)

Autor:
Say Hood T. Chiu Acosta

Tutor:
Prof. Eric Gamess

El protocolo IPv6 define la interacción entre los nodos que residen en un mismo enlace a través del proceso de descubrimiento de vecinos (Neighbor Discovery). Los mensajes intercambiados durante este descubrimiento incluyen información importante acerca de la red y pueden dar lugar a diversos ataques que comprometen el correcto funcionamiento de los nodos del enlace.

En este Trabajo Especial de Grado se desarrolló una aplicación que permite asegurar los mensajes enviados por los nodos durante el proceso de descubrimiento de vecinos al utilizar el protocolo SEND. Además, se desarrolló una herramienta que permite generar y verificar direcciones CGA (Cryptographically Generated Addresses).

El trabajo se realizó en base a las teorías sobre los protocolos y mecanismos de seguridad que garantizan la protección de los mensajes de Neighbor Discovery. Durante la implementación de la aplicación, se utilizó la metodología de desarrollo de software llamada *enfoque en cascada* que contempla las fases de: análisis, desarrollo, implementación y pruebas necesarias para comprobar el funcionamiento de la aplicación SEND desarrollada.

Es de hacer notar que el alcance de esta implementación de SEND sólo abarcó el intercambio de mensajes entre los hosts del enlace, es decir que no involucra el descubrimiento de los routers que puedan estar presentes. Sin embargo, podrá servir como referente teórico a futuros trabajos en el área, en la cual puedan incluir el proceso de Router Discovery y otras funcionalidades que vayan surgiendo.

Como producto final se incluyeron el código fuente, la documentación de la aplicación y el manual de usuario, necesarios para comprender el correcto funcionamiento de la aplicación SEND, así como un *virtual appliance* que facilita la ejecución de pruebas para un posterior análisis.

Palabras claves: IPv6, Neighbor Discovery, Criptografía, Firma Digital, Seguridad, CGA.

Tabla de Contenido

Índice de Figuras **13**
Índice de Tablas **15**
Introducción **17**
1. El Problema **19**
1.1 Planteamiento del Problema 19
1.2 Objetivos 19
1.2.1 Objetivo General 19
1.2.2 Objetivos Específicos 19
1.3 Justificación 20
1.4 Alcance 20
2. Marco Teórico **21**
2.1 Neighbor Discovery 21
2.1.1 Mensajes 22
2.1.2 Opciones 29
2.1.3 Proceso de Descubrimiento 30
2.1.4 Vulnerabilidades en Neighbor Discovery 34
2.2 Secure Neighbor Discovery 37
2.2.1 Mensajes 37
2.2.2 Opciones 40
2.2.3 Authorization Delegation Discovery (ADD) 50
2.2.4 La Transición 55
2.2.5 Consideraciones de Seguridad 56
2.2.6 Resumen de Tablas Definidas por el Protocolo SEND 57
2.3 DoCoMo's Open Source SEND Project 59
2.3.1 Descripción 59
2.3.2 Ventajas 59
2.3.3 Desventajas 62
3. Marco Metodológico **63**
3.1 Metodología de Desarrollo 63
3.1.1 Análisis de los Requerimientos de la Aplicación 63
3.1.2 Diseño de la Aplicación 64
3.1.3 Desarrollo e Implementación de la Aplicación 66
3.1.4 Integración y Pruebas 66
4. Marco Aplicativo **67**
4.1 Fase de Diseño 67
4.2 Fase de Desarrollo 76
4.2.1 Requerimientos Mínimos del Sistema 76
4.2.2 Puesta en Marcha 77
4.3 Fase de Pruebas 78
5. Conclusiones y Trabajos Futuros **85**
Apéndice A: Manual de Usuario **87**
Introducción 87
Requisitos Previos 87
Puesta en Marcha de la Aplicación 87
Configuración 88
Ejecutando "sendapp.jar" 89
Limitaciones 90

Apéndice B: Virtual Appliance 91
Apéndice C: Generando una Dirección CGA 93
Apéndice D: Analizando un Paquete con Opciones SEND 95
Referencias Bibliográficas 101
Glosario de Términos 103

Índice de Figuras

Figura 2.1: Estructura de un paquete IPv6 22
Figura 2.2: Estructura de la cabecera IPv6 22
Figura 2.3: Estructura del mensaje ICMPv6 23
Figura 2.4: Estructura del mensaje ND 23
Figura 2.5: Estructura Router Solicitation 24
Figura 2.6: Estructura Router Advertisement 25
Figura 2.7: Estructura Neighbor Solicitation 26
Figura 2.8: Estructura Neighbor Advertisement 27
Figura 2.9: Estructura Redirect 28
Figura 2.10: Estructura campo Options 29
Figura 2.11: Router Discovery 31
Figura 2.12: Redirect 32
Figura 2.13: Address Resolution 33
Figura 2.14: Duplicate Address Detection 34
Figura 2.15: Estructura CPS 38
Figura 2.16: Estructura CPA 39
Figura 2.17: CGA 40
Figura 2.18: CGA Parameter Data Structure 41
Figura 2.19: Estructura CGA Option 43
Figura 2.20: Paquete IPv6 con CGA Option 44
Figura 2.21: Estructura RSA Option 44
Figura 2.22: RSA Digital Signature contents 46
Figura 2.23: Estructura Timestamp Option 46
Figura 2.24: Estructura Nonce Option 48
Figura 2.25: Estructura Trust Anchor Option 49
Figura 2.26: Estructura Certificate Option 49
Figura 2.27: Certification Path 50
Figura 2.28: Delegate Authority Model 51
Figura 2.29: Ejemplo Camino de Certificación 53
Figura 2.30: Ejemplo cgatool 59
Figura 2.31: Captura de un mensaje NS con opciones SEND 60
Figura 2.32: Captura de un mensaje NA con opciones SEND 60
Figura 2.33: Captura de un mensaje CPA 61
Figura 2.34: Captura de un mensaje RA que utiliza opciones SEND 61
Figura 3.1: Arquitectura de la aplicación 65
Figura 4.1: Diagrama de clase NDMessage 67
Figura 4.2: Diagrama de clase SENDOPT 68
Figura 4.3: Diagrama de clase CGA 69
Figura 4.4: Cálculo del Hash2 en CGA 70
Figura 4.5: Cálculo del Hash1 en CGA 70
Figura 4.7: Diagrama de clase RSASignature 71
Figura 4.6: Dirección CGA 71
Figura 4.8: Diagrama de clase Nonce 72
Figura 4.9: Diagrama de clase TimeStamp 73
Figura 4.10: Diagrama de clase TimestampCache 74
Figura 4.11: Diagrama de clase Monitor 74
Figura 4.12: Diagrama de clase KeyManagement 75

Figura 4.13: Diagrama de clases Files y Util 75
Figura 4.14: Diagrama de clase NetInterfaces 76
Figura 4.15: Diagramas de clases Configuration y Sendapp 76
Figura 4.16: Topología de la red de pruebas 78
Figura 4.17: ping entre dos hosts con “sendapp” – (CP1) 79
Figura 4.18: DAD – (CP3) 81
Figura 4.19: ping entre un host “sendapp” y un host “sendd” – (CP6) 83

Índice de Tablas

Tabla 2.1: ICMPv6 Type para mensajes ND 23
Tabla 2.2: Options Type para ND 29
Tabla 2.3: ICMPv6 Type para mensajes SEND 38
Tabla 2.4: Options Type para SEND 40
Tabla 2.5: Resumen mensaje ND 57
Tabla 2.6: Resumen opciones ND 57
Tabla 2.7: Constantes SEND 58
Tabla 2.8: Variables SEND 58
Tabla 4.1: Características de los hosts de pruebas 78

Introducción

Cuando se diseñó el actual protocolo de red de Internet (IPv4), no se pensó en el crecimiento exponencial que se ha experimentado en los últimos años; cada vez hay más usuarios y el espacio de direcciones que provee IPv4 es insuficiente.

Para tratar de suplir las carencias que actualmente tiene IPv4, nació una versión del protocolo llamada IPv6, la cual a pesar de tener ya varios años creada, es ahora cuando ha tomado importancia.

Cabe destacar que una de las características más poderosas y flexibles del protocolo IPv6 es el proceso de descubrimiento de vecinos (Neighbor Discovery); sin embargo, la flexibilidad y la seguridad son justamente dos requerimientos opuestos.

El protocolo ND (Neighbor Discovery) es el responsable de encontrar otros nodos (hosts y routers) dentro del mismo segmento de red, pero las especificaciones [1] indican que está propenso a sufrir diferentes ataques que podrían hacer que el flujo de paquetes IPv6 alcancen un destino inesperado [7].

No obstante, el principal detalle durante el proceso de descubrimiento de vecinos ocurre cuando un host se une a la red, ya que no sabe quiénes son los nodos que allí se encuentran y por lo tanto, es imposible juzgar la integridad de ellos. Por tal razón, se recomienda el uso de IPsec o SEND (Secure Neighbor Discovery) como mecanismos de seguridad [1] para reducir las amenazas en las que se ve envuelto el protocolo ND.

Con respecto a IPsec como mecanismo de seguridad nativo de IPv6, este no provee una protección de manera automática para el proceso de autoconfiguración de los hosts. Por tal motivo, son pocas las aplicaciones que se apoyan sobre la implementación de IPsec, dado que el manejo de las asociaciones de seguridad tiene que hacerse de forma manual y en consecuencia su uso se torna impracticable debido a la complejidad que acarrea su configuración [9].

Por otra parte, con el protocolo SEND como mecanismo de seguridad se puede ofrecer protección a los mensajes ND haciendo uso de la criptografía asimétrica, logrando brindar integridad y autenticación a los paquetes intercambiados durante el proceso ND.

El presente trabajo se estructura en cinco capítulos; los cuales se detallan a continuación: En el Capítulo 1, se expone la problemática, objetivos, justificación y alcance de la investigación. El Capítulo 2 contiene las bases teóricas que explican en detalle los protocolos, mecanismos y consideraciones de seguridad sobre los que se fundamenta el desarrollo de la aplicación. El Capítulo 3 describe la metodología utilizada y otros aspectos relevantes tomados en cuenta para la implementación. El Capítulo 4 explica en detalle el proceso de implementación. Finalmente, el Capítulo 5 presenta las conclusiones y trabajos futuros.

1. El Problema

1.1 Planteamiento del Problema

Actualmente los fabricantes de dispositivos de redes y los desarrolladores de sistemas operativos para computadores, no incluyen la funcionalidad del protocolo SEND dentro de sus productos. La única implementación de SEND disponible fue desarrollada por miembros pertenecientes al grupo de trabajo que realizó las especificaciones del protocolo SEND[1].

Existe también un proyecto basado en Java de nombre JSend[2], pero a la fecha, no hay ningún código disponible.

Por otra parte, el ETSI (European Telecommunications Standards Institute) ha producido un marco para el desarrollo de pruebas IPv6 que incluye procedimientos, librerías y suites de pruebas sobre las áreas de movilidad, seguridad, transición y calidad de servicio; pero hasta la fecha no ha incluido entre sus planes una implementación de SEND[3].

Es por ello que se desea desarrollar una aplicación que permita implementar las especificaciones hechas en el protocolo SEND y contribuir a llenar un poco el vacío existente. La aplicación estará bajo licencia de software libre y podrá operar en sistemas operativos Unix que tengan soporte en su núcleo para IPv6.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar una aplicación que permita implementar parte del protocolo Secure Neighbor Discovery en una red IPv6.

1.2.2 Objetivos Específicos

- Analizar los protocolos y algoritmos que serán cubiertos por la aplicación.
- Diseñar la aplicación.
- Implementar la aplicación.
- Realizar pruebas con la finalidad de constatar posibles fallas o su correcto funcionamiento.
- Elaborar la documentación de la aplicación para su uso y posterior mejora.

[1] http://www.docomolabs-usa.com/lab_opensource.html
[2] http://sourceforge.net/projects/jsend
[3] http://aui.es/index.php?body=mip_article_ponencia&id_article=1359

1.3 Justificación

Debido a los diferentes ataques a los que está expuesto el protocolo Neighbor Discovery y la falta de desarrollo e implementación del protocolo SEND por parte de los fabricantes y desarrolladores del área, se hace necesario el desarrollo de una aplicación que permita implementar las especificaciones descritas en el protocolo Secure Neighbor Discovery, y que brinde por lo tanto la seguridad que las redes IPv6 necesitan; llenando en este sentido el vacío que existe actualmente en cuanto a las implementaciones de dicho protocolo.

1.4 Alcance

La aplicación desarrollada es capaz de establecer el mecanismo de seguridad deseado durante el proceso de descubrimiento de los nodos de una red IPv6, a través del protocolo SEND tal como lo indican las especificaciones [8]. Además, es versátil para la integración de nuevas opciones que puedan surgir y funcionar en las plataformas Unix que tengan soporte para IPv6.

Cabe destacar que dicha aplicación no abarca el proceso de descubrimiento de los routers para los hosts que puedan estar presentes en el enlace local.

2. Marco Teórico

2.1 Neighbor Discovery

Neighbor Discovery es un protocolo de IPv6 que comprende un conjunto de mensajes y procesos, con el cual un nodo que se incorpora a una red descubre la presencia de otros nodos en su mismo enlace [1].

ND reemplaza al protocolo ARP (Address Resolution Protocol) e incorpora las funcionalidades de otros protocolos de IPv4, como ICMP *Router Discovery* e ICMP *Redirect*. También se ocupa de mantener la consistencia de la cache de vecinos, donde se almacena la información relativa al contexto de la red a la que está directamente conectado un nodo.

Este protocolo define una serie de mecanismos y procesos usados para resolver el problema relacionado con la interacción entre los nodos de un mismo enlace. A continuación se describen dichos mecanismos y procesos:

- ***Router Discovery***: permite a los hosts localizar los routers que residen en el mismo enlace.
- ***Prefix Discovery***: proceso por el cual un host descubre los prefijos de red del enlace. Su objetivo es poder distinguir entre máquinas directamente conectadas y máquinas que requieren el enrutamiento de un router para poder comunicarse.
- ***Parameter Discovery***: permite a un host descubrir parámetros adicionales del enlace como el MTU o parámetros de Internet como el Hop Limit usado en los paquetes salientes.
- ***Address Autoconfiguration***: introduce los mecanismos necesarios para permitir que los hosts se configuren con la autoconfiguración de tipo stateless [2]. Ya no es prioritaria la utilización de DHCP (autoconfiguración de tipo stateful) para la obtención automática de direcciones.
- ***Address Resolution***: permite resolver una dirección IPv6 a una dirección MAC, es decir, el equivalente al protocolo ARP en IPv4.
- ***Next-Hop Determination***: proceso mediante el cual un nodo determina la dirección IPv6 del vecino que se encargará de reenviar el paquete hacia la dirección de destino.
- ***Neighbor Unreachability Detection***: permite determinar cuando un nodo ya no está disponible dentro del enlace.
- ***Duplicate Address Detection***: proceso a través del cual un nodo determina si un nodo vecino está o no utilizando la dirección IPv6 solicitada. Se usa para prevenir las colisiones de las direcciones IPv6 durante el proceso de autoconfiguración.
- ***Redirect Function***: proceso por el cual un router informa a un host una mejor ruta para llegar a un determinado destino.

2.1.1 Mensajes

Un paquete IPv6 está compuesto principalmente en tres partes: la cabecera IPv6, las cabeceras de extensión y los datos de la capa superior. En la **¡Error! No se encuentra el origen de la referencia.** (tomada de [14]) se muestra la estructura de un paquete IPv6.

Figura 2.1: Estructura de un paquete IPv6

La *cabecera IPv6* constituye los primeros 40 bytes de un paquete IPv6 y contiene una variedad de campos, como se puede observar en la Figura 2.2 (tomada de [3]).

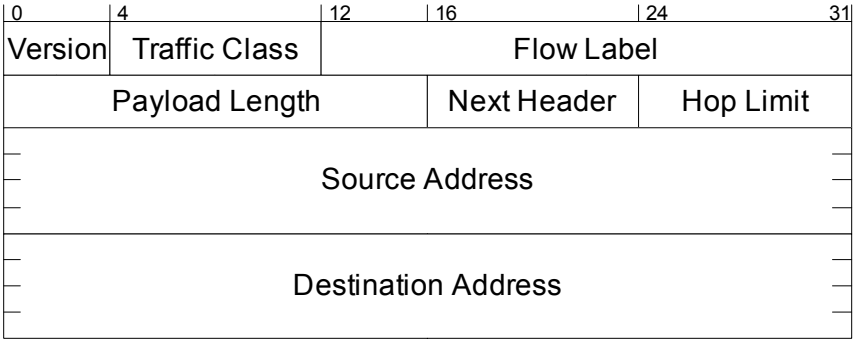

Figura 2.2: Estructura de la cabecera IPv6

Las *cabeceras de extensión IPv6* son opcionales y usadas para ir añadiendo funcionalidades de forma paulatina, específicamente opciones de seguridad, enrutamiento, y fragmentación. En un paquete IPv6, si existen una o más cabeceras de extensión, las mismas deben estar previamente identificadas por el campo N*ext Header* de la cabecera que la precede. Actualmente existen definidas seis cabeceras de extensión IPv6 y deben ubicarse en un orden específico [3].

Los *datos de la capa superior* consisten típicamente en una cabecera propia del protocolo que se está transportando (ICMPv6, UDP, TCP, etc.) y su carga útil. Los mensajes intercambiados en el protocolo ND son de tipo ICMPv6 (Internet Control Message Protocol version 6).

Para llevar a cabo el proceso de descubrimiento, ND define cinco tipos de paquetes ICMPv6: Router Solicitation (RS), Router Advertisement (RA), Neighbor Solicitation (NS), Neighbor Advertisement (NA) y Redirect. ICMPv6 es considerado un protocolo de la capa superior [4] ya que los mensajes van encapsulados en paquetes IPv6 y el valor del campo *Next Header* de la cabecera IPv6 utilizado para identificar dicho protocolo es 58.

Al igual que el protocolo ICMP para IPv4, ICMPv6 también es usado para reportar mensajes de error y mensajes informativos. La Figura 2.3 (tomada de [4]) muestra la estructura de los mensajes ICMPv6.

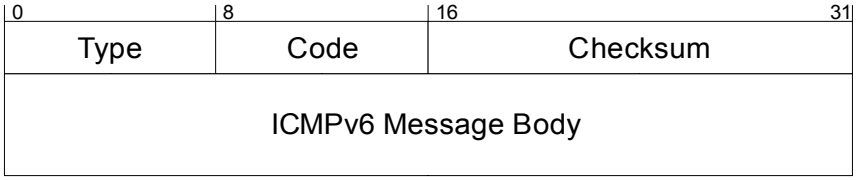

Figura 2.3: Estructura del mensaje ICMPv6

Los primeros 32 bits de un paquete ICMPv6 representan la cabecera ICMPv6 y son iguales tanto para los mensajes de error como para los mensajes de información.

Los ocho bits del campo ***Type*** indican el tipo de mensaje que va a ser transportado. Los primeros 128 valores son mensajes de error, mientras que los 128 restantes son mensajes de información.

El campo ***Code*** es utilizado para diferenciar entre múltiples mensajes de un mismo tipo. El campo ***Checksum*** es usado para verificar la integridad de los mensajes ICMPv6.

Dado que el protocolo ND hace uso de los mensajes ICMPv6, la Figura 2.4 (tomada de [14]) muestra la estructura de un mensaje ND:

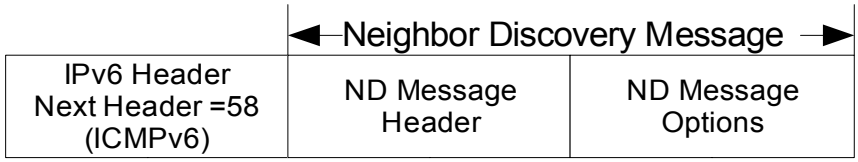

Figura 2.4: Estructura del mensaje ND

La cabecera ND está compuesta por los primeros 32 bits de la cabecera ICMPv6 y los datos previos a las opciones del mensaje ND que se quiere transportar. Los valores asociados al campo *Type* del encabezado ICMPv6 definidos en el protocolo ND [1] se muestran en la Tabla 2.1.

ICMPv6 Type	ND Message
133	Router Solicitation (RS)
134	Router Advertisement (RA)
135	Neighbor Solicitation (NS)
136	Neighbor Advertisement (NA)
137	Redirect

Tabla 2.1: ICMPv6 Type para mensajes ND

Para asegurar que todos los mensajes ND provienen del mismo enlace local, el valor de campo *Hop Limit* de la cabecera IPv6 (ver Figura 2.2) es 255 y debe ser comprobado por el receptor del mensaje. Si el valor del campo *Hop Limit* es diferente a 255, el mensaje debe ser desechado.

2.1.1.1 Router Solicitation

Los mensajes RS son enviados por los hosts IPv6 para descubrir la presencia de routers IPv6 en el enlace. Este tipo de mensajes hace que los routers envíen como respuesta un mensaje RA, sin necesidad de esperar a que expire el temporizador del router para que sean enviados. La Figura 2.5 (tomada de [1]) muestra la estructura de un RS.

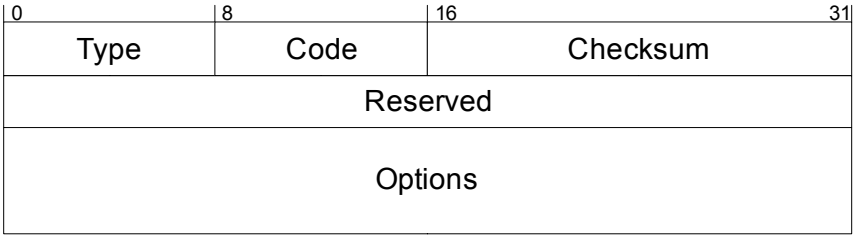

Figura 2.5: Estructura Router Solicitation

Los campos de la cabecera IPv6 deben contener la siguiente información:

Source Address: la dirección IPv6 asignada a la interfaz o la dirección no especificada (::) cuando no se le ha asignado alguna dirección.

Destination Address: la dirección IPv6 multicast de All-Routers (FF02::2).

Hop Limit: el valor debe ser 255.

A nivel de ICMPv6, los campos deben contener la siguiente información:

Type: debe ser 133.

Code: debe ser 0.

Checksum: el valor del ICMPv6 Checksum.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Options: debe ser Source Link-Layer Address.

2.1.1.2 Router Advertisement

Los routers envían mensajes RA en intervalos regulares de tiempo (RA no solicitado) o en respuesta a un mensaje RS (RA solicitado). Los mensajes RA pueden contener la información requerida por los hosts para determinar el prefijo de red, MTU y rutas específicas. En la Figura 2.6 (tomada de [1]) se observa la estructura de un mensaje RA.

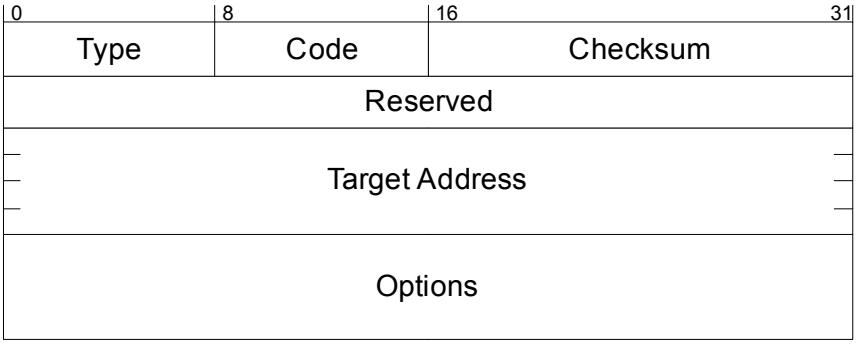

Figura 2.6: Estructura Router Advertisement

Los campos de la cabecera IPv6 deben contener la siguiente información:

Source Address: la dirección IPv6 unicast de tipo Link-Local asignada a la interfaz.

Destination Address: típicamente la dirección IPv6 del nodo que emitió el RS o la dirección multicast de All-Nodes (FF02::1).

Hop Limit: el valor debe ser 255.

A nivel de ICMPv6, los campos deben contener la siguiente información:

Type: debe ser 134.

Code: debe ser 0.

Checksum: el valor del ICMPv6 Checksum.

Cur Hop Limit: indica a los hosts el valor por defecto que deben usar en el campo *Hop Limit* de la cabecera IPv6. Un valor igual a cero indica que el router no especifica el valor por defecto del campo *Hop Limit*.

Managed Address Configuration flag (M): cuando el bit está en uno (1), indica que el host debe usar un protocolo de autoconfiguración de tipo stateful (DHCPv6) para obtener direcciones IPv6 adicionales a las obtenidas durante la autoconfiguración de tipo stateless.

Other Stateful Configuration flag (O): cuando el bit está en uno (1), indica que los hosts deben obtener información adicional por la autoconfiguración de tipo stateful como por ejemplo la lista de servidores DNS.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Router Lifetime: en este campo, el valor es un número entero que representa el tiempo en el que el router será router por defecto para los hosts. El máximo valor que puede tomar este campo es 65.535 segundos, aproximadamente 18 horas. Si el valor del campo es cero, significa que el router no es el router por defecto y por lo tanto, no debe aparecer en la lista de los routers por defecto de los hosts.

Reachable Time: indica la cantidad de tiempo en milisegundos que un nodo puede considerar a un vecino como disponible, después de haber recibido una

prueba de accesibilidad. Un valor igual a cero indica que el router no hace recomendación para este campo.

Retransmission Timer: indica el tiempo en milisegundos que debe transcurrir entre dos mensajes NS consecutivos. Un valor igual a cero indica que el router no hace recomendación para este campo.

Options: contiene Prefix Information, MTU, Source Link-Layer Address.

2.1.1.3 Neighbor Solicitation

Los hosts envían estos mensajes para descubrir la dirección de capa de enlace de sus vecinos (dirección MAC) o para verificar si un nodo sigue activo (es alcanzable). Estos mensajes pueden ser de tipo multicast para la resolución de direcciones o unicast cuando la accesibilidad o conectividad de un nodo está siendo verificada. La Figura 2.7 (tomada de [1]) muestra la estructura de un mensaje NS.

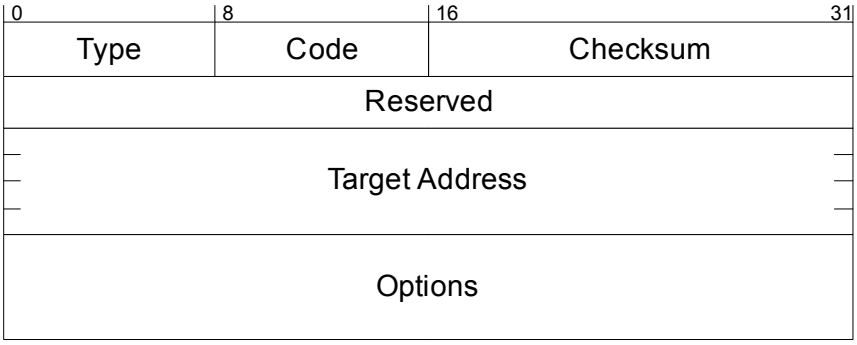

Figura 2.7: Estructura Neighbor Solicitation

Los campos de la cabecera IPv6 deben contener la siguiente información:

Source Address: la dirección IPv6 unicast de tipo Link-Local del emisor o la dirección no especificada (::).

Destination Address: cuando el mensaje generado es para determinar la dirección MAC de un host, el mensaje NS es de tipo multicast y la dirección de destino será del tipo Solicited Node. Un mensaje NS es de tipo unicast cuando se desea verificar la conectividad de un determinado nodo, en este caso, la dirección de destino será directamente la que corresponde a dicho nodo.

Hop Limit: el valor debe ser 255.

A nivel de ICMPv6, los campos deben contener la siguiente información:

Type: debe ser 135.

Code: debe ser 0.

Checksum: el valor del ICMPv6 Checksum.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Target Address: es la dirección del destino solicitado, no debe ser una dirección de tipo multicast.

Options: contiene la opción Source link-Layer address.

2.1.1.4 Neighbor Advertisement

Un nodo envía un mensaje NA en respuesta a un mensaje NS o para propagar nueva información a los nodos vecinos, indicando el cambio ocurrido en la dirección de capa de enlace (dirección MAC) de un nodo específico dentro del segmento. La Figura 2.8 (tomada) de [1] muestra la estructura de un mensaje NA.

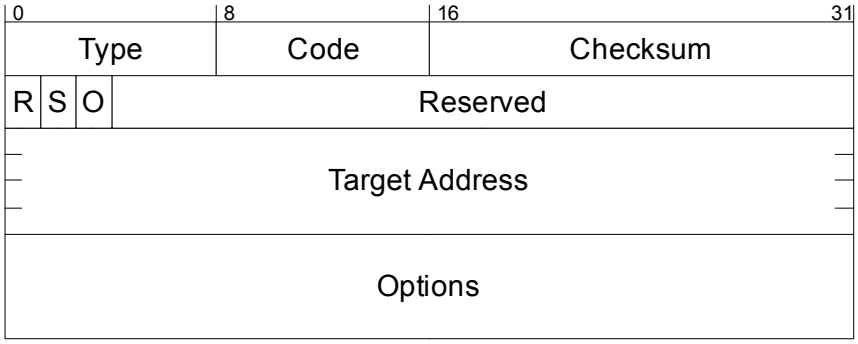

Figura 2.8: Estructura Neighbor Advertisement

Los campos de la cabecera IPv6 deben contener la siguiente información:

Source Address: la dirección IPv6 unicast de tipo Link-Local asignada a la interfaz.

Destination Address: en respuesta a un mensaje NS, la dirección de destino será la dirección unicast del remitente. Para un mensaje NA no solicitado, la dirección de destino será la dirección multicast de All-Nodes (FF02::1).

Hop Limit: el valor debe ser 255.

A nivel de ICMPv6, los campos deben contener la siguiente información:

Type: debe ser 136.

Code: debe ser 0.

Checksum: el valor del ICMPv6 Checksum.

Router Flag (R): si el bit está en uno (1), es porque el remitente es un router. Cuando el bit está en cero indica que es un host. Esta bandera es usada por el algoritmo NUD para determinar cuando un router cambia de rol, es decir, el router pasa a ser un host.

Solicited Flag (S): cuando el bit está en uno (1), indica que el mensaje NA es enviado en respuesta a un mensaje NS. Este bit está en cero en los mensajes NA de tipo multicast.

Override Flag (O): cuando el bit está en uno (1), indica al destino que se debe sobrescribir la entrada de la cache de vecinos con la dirección enviada en la opción Target Link-Layer Address. Si el bit está en cero, el destino debe asegurarse de actualizar su cache con una nueva entrada, cuando la dirección de la opción no es conocida.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Target Address: para los mensajes NA solicitados, la dirección corresponde con la dirección del campo *Target Address* previamente solicitado en el mensaje NS. Para los mensajes NA no solicitados, la dirección será la dirección del nodo que haya cambiado su rol o la nueva dirección física (dirección MAC). Nunca debe ser una dirección de tipo multicast.

Options: debe ser la opción Target Link-Layer Address.

2.1.1.5 Redirect

Estos mensajes son generados solamente por los routers para informar a un host de un mejor camino hacia una red específica. La Figura 2.9 (tomada de [1]) muestra la estructura de un mensaje Redirect.

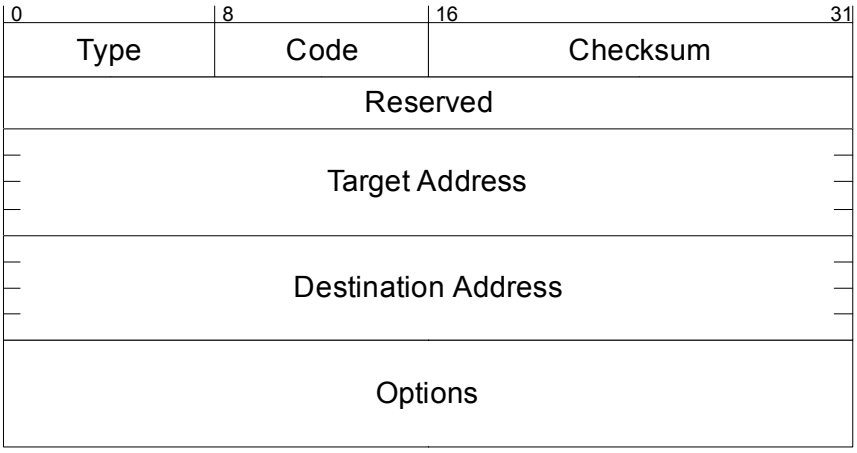

Figura 2.9: Estructura Redirect

Los campos de la cabecera IPv6 deben contener la siguiente información:

Source Address: la dirección IPv6 unicast de tipo link-local asignada a la interfaz.

Destination Address: la dirección de la fuente que provocó la redirección.

Hop Limit: el valor debe ser 255.

A nivel de ICMPv6, los campos deben contener la siguiente información:

Type: debe ser 137.

Code: debe ser 0.

Checksum: el valor del ICMPv6 Checksum.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Target Address: indica la nueva dirección a donde deberán ser enviados los paquetes para que puedan alcanzar su destino. Si los paquetes deben viajar

más allá del enlace local, la dirección del campo *Target Address* será la del router de frontera. Si los paquetes no salen del enlace, la dirección será la misma del campo *Destination Address*.

Destination Address: contiene la dirección de destino del paquete que hizo que el router enviara un mensaje de tipo Redirect.

Options: pueden ser Target Link-Layer Address o Redirected Header

2.1.2 Opciones

Los mensajes del protocolo ND contienen en su parte inferior un campo llamado Options, el cual está formado por una estructura denominada TLV (Type Length Value) tal como se muestra en la Figura 2.10.

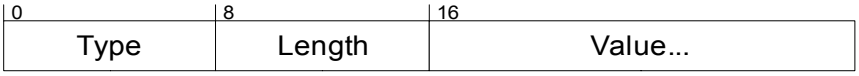

Figura 2.10: Estructura campo Options

El campo ***Type*** representa los primeros ocho bits de las opciones en un mensaje ND e indica el tipo de la opción que se transporta en dicho mensaje [1][5][6].

Entre los principales tipos de opciones se encuentran: Source Link-Layer Address, Target Link-Layer Address, Prefix Information, Redirected Header, y Maximum Transfer Unit. En la Tabla 2.2 se especifican los valores del campo *Type* en función de la opción que representa.

Type	Options Name
1	Source Link-Layer Address
2	Target Link-Layer Address
3	Prefix Information
4	Redirected Header
5	Maximum Transmission Unit (MTU)

Tabla 2.2: Options Type para ND

Los ocho bits del campo ***Length*** indican la longitud de la opción en bloques de ocho bytes. Los mensajes ND con opciones que contienen el valor *Length* igual a cero deben ser desechados.

El campo ***Value*** contiene los datos de la opción. Este campo es de tamaño variable.

2.1.2.1 Source Link-Layer Address

Esta opción es incluida en los mensajes NS, RS y RA para indicar la dirección MAC del remitente del mensaje. Si la dirección fuente del mensaje ND es la dirección no especificada (::), esta opción no es incluida dentro del mensaje.

2.1.2.2 Target Link-Layer Address

Con esta opción se indica la dirección MAC del nodo vecino a donde serán dirigidos los paquetes. Los mensajes NA y Redirect hacen uso de esta opción.

2.1.2.3 Prefix Information

Esta opción es enviada en los mensajes RA para indicar los prefijos a usar en la autoconfiguración. Puede haber múltiples opciones de tipo Prefix Information en un mensaje RA, indicando múltiples prefijos que pueden ser utilizados por los nodos para crear sus direcciones IPv6.

2.1.2.4 Redirected Header

Esta opción sólo es usada en los mensajes Redirect para especificar el paquete IPv6 que hizo que el router enviara un mensaje Redirect. La opción puede contener toda o una parte del paquete que fue enviado inicialmente, dependiendo del tamaño del mismo. La cantidad del paquete original que es incluido, debe ser tal que el mensaje Redirect no sea mayor a los 1280 bytes de longitud.

2.1.2.5 Maximum Transmission Unit (MTU)

Esta opción es enviada en los mensajes RA para indicar el MTU IPv6 usado en el enlace. Típicamente esta opción es incluida en los enlaces que tienen diferentes MTU o en ambientes con múltiples tecnologías en la capa de enlace.

2.1.3 Proceso de Descubrimiento

Como se describió anteriormente, el proceso de descubrimiento de vecinos consiste en una serie de mecanismos que permiten establecer las relaciones de los nodos dentro del enlace. A continuación se muestra una serie de ejemplos que ilustran algunos de dichos mecanismos [14].

Ejemplo 1:
Router Discovery: el proceso de descubrimiento de los routers en IPv6 es similar al ICMP Router Discovery para IPv4, ya que permite localizar los router vecinos que residen dentro del enlace, aprender los prefijos de red y los parámetros de configuración relacionados con la autoconfiguración. Con el descubrimiento del prefijo de red, los hosts conocen los diferentes prefijos que se encuentran en el enlace y con el cual puede comunicarse con otros nodos, sin necesidad de atravesar un router de frontera.

Los mensajes que conforman el descubrimiento del router son los mensajes RA y RS. La Figura 2.11 (tomada de [14]) muestra el intercambio realizado durante el proceso de Router Discovery, cuando la dirección de la fuente del mensaje RS es la no especificada (::).

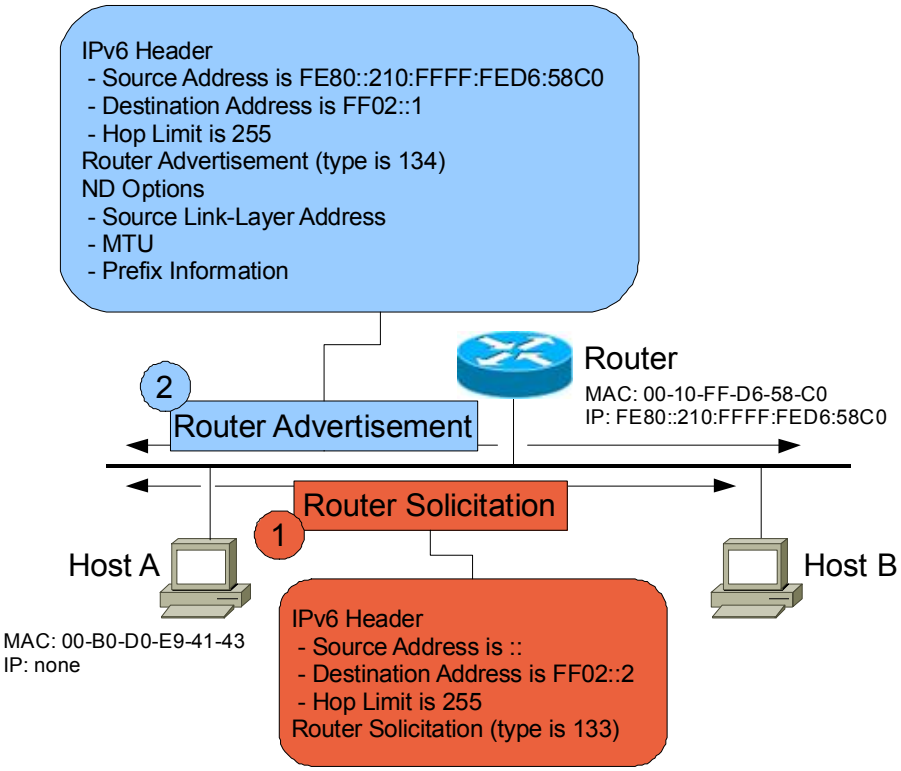

Figura 2.11: Router Discovery

A diferencia del ICMP Router Discovery para IPv4, en el Router Discovery de IPv6 cuando un router por defecto deja de serlo, los hosts pueden detectar el cambio y escoger uno nuevo de la lista de router por defecto gracias al algoritmo NUD (Neighbor Unreachability Detection), lo cual evita que los hosts tengan que esperar a que termine el tiempo de vida de su router por defecto para poder descubrir otro.

Ejemplo 2:
Redirect Function: los routers envían mensajes Redirect para informar a los hosts un mejor camino para enviar los paquetes. Existen dos casos por los cuales es necesario que un router origine este tipo de mensajes:

- Un router informa a un host que está disponible una dirección IPv6 de un router que se encuentra más cerca del destino final del paquete enviado por el host. Esto ocurre cuando hay más de un router dentro del enlace y el router por defecto elegido por el host puede no ser el más cercano al destinatario.
- Un router informa a un host que el destinatario del paquete se encuentra dentro del enlace, es decir, es un nodo vecino. Esto puede ocurrir cuando no se incluye el prefijo del destinatario del paquete en la lista de prefijos anunciados por el router, lo que provoca que el paquete sea enviado al router por defecto, ya que no existe una entrada en la lista que se pueda cotejar.

La Figura 2.12 (tomada de [14]) muestra un ejemplo de un mensaje Redirect.

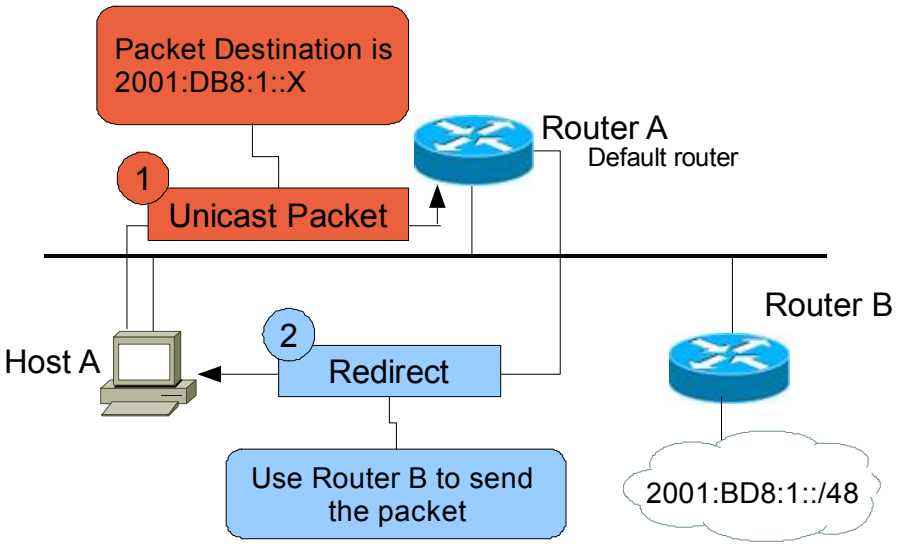

Figura 2.12: Redirect

Ejemplo 3:
Address Resolution: el proceso de resolución de direcciones IPv6 consiste en el intercambio de mensajes NS y NA. Al igual que en el protocolo ARP de IPv4, la idea es resolver una dirección IPv6 a una dirección MAC.

Un host envía un mensaje NS de tipo multicast, donde la dirección de destino será del tipo Solicited-Node derivada de la dirección IPv6 del nodo solicitado. También se incluye la dirección MAC del remitente del mensaje NS dentro de la opción Source Link-Layer Address, de modo que el receptor del mensaje pueda actualizar su cache de vecinos con la dirección fuente y la dirección MAC del emisor.

El receptor del mensaje NS envía como respuesta un mensaje NA incluyendo su dirección MAC en la opción de Target Link-Layer Address, de esta forma el remitente del mensaje NS podrá actualizar su cache de vecinos y los paquetes posteriores entre los nodos involucrados serán de tipo unicast. La Figura 2.13 (tomada de [14]) ilustra el proceso de resolución de dirección.

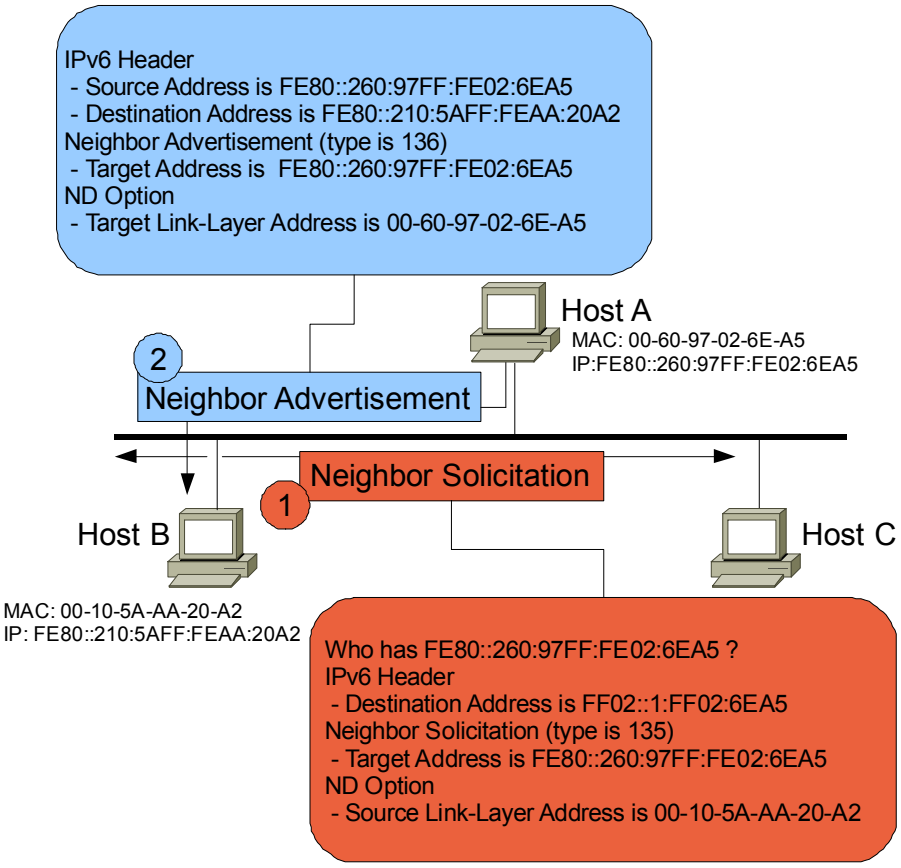

Figura 2.13: Address Resolution

Ejemplo 4:
Duplicate Address Detection (DAD)*:* este proceso es similar al *gratuitous* ARP en las redes IPv4. Cuando un nodo se une a un enlace y pretende usar su dirección IPv6 o simplemente quiere hacer un cambio de la misma, el nodo debe ejecutar un proceso de detección de duplicidad de la dirección IPv6 y así evitar que se produzca una colisión de la dirección.

Para detectar si algún nodo está haciendo uso de la dirección IPv6 solicitada, se usan los mensajes de NS y NA. Los mensajes NS enviados durante el proceso DAD llevan en el campo *Source Address* de la cabecera IPv6 (ver Figura 2.2) la dirección no especificada (::), ya que no pueden usar la dirección solicitada hasta que se determine que no está siendo usada por otro nodo dentro del enlace.

Los mensajes NA en respuesta a mensajes NS, son de tipo multicast y la dirección de destino en el campo *Destination Address* de la cabecera IPv6 es de tipo All-Nodes (FF02::1).

Si el nodo que está ejecutando el DAD no recibe una respuesta (un mensaje NA) él será libre de usar la dirección IPv6 solicitada; en caso contrario deberá intentar con otra dirección ya que la dirección solicitada está siendo usada por algún nodo dentro del enlace. La Figura 2.14 muestra el proceso DAD

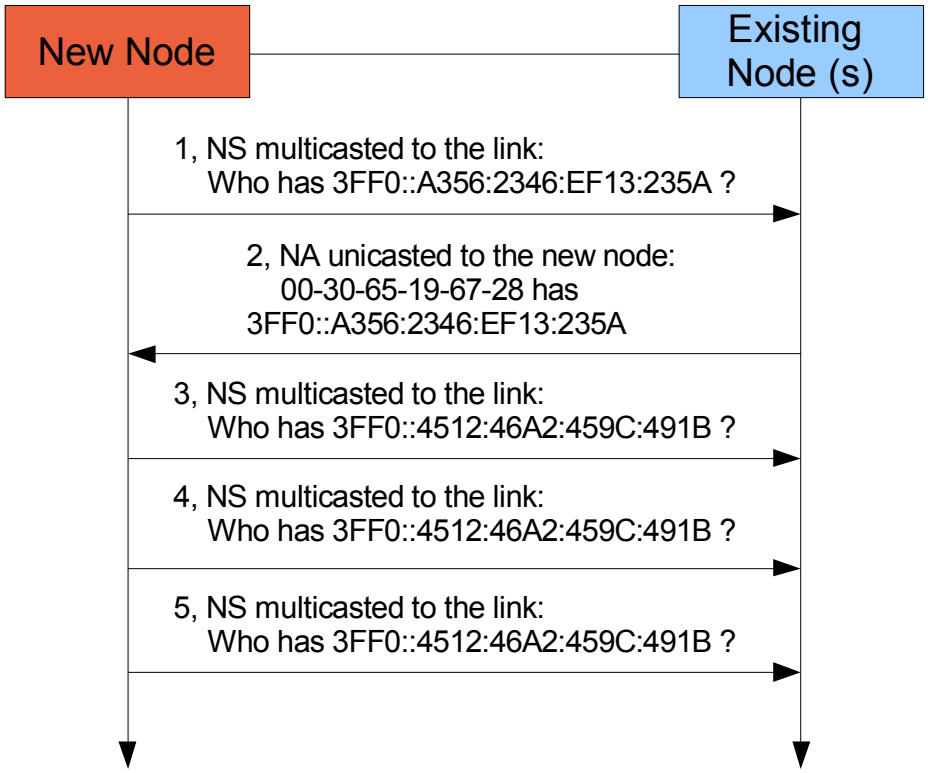

Figura 2.14: Duplicate Address Detection

2.1.4 Vulnerabilidades en Neighbor Discovery

ND está propenso a sufrir diferentes ataques que pueden causar que los paquetes IPv6 tomen un rumbo inesperado. Estos ataques pueden ser usados para causar una denegación de servicio, interceptar y opcionalmente modificar los paquetes destinados a otros nodos del enlace.

Los ataques contra el protocolo ND pueden ser clasificados en tres tipos:

- ***Impersonation/Spoofing***: con este tipo de ataque un nodo puede hacerse pasar por otro, ya que no hay control de las direcciones MACs y pueden ser fácilmente cambiadas.
- ***DoS (Denial of Service)****:* con este tipo de ataque un nodo malintencionado impide la comunicación entre la víctima y los demás nodos del enlace o una dirección de destino específica.
- ***Redirection***: con este método de ataque, el nodo malintencionado puede dirigir o desviar los paquetes a otro nodo del enlace.

Entre los potenciales ataques más comunes [7] que amenazan la seguridad del enlace durante el intercambio de mensajes en el proceso ND se tienen:

- **Neighbor Solicitation / Neighbor Advertisement Spoofing**:
  - El atacante puede hacer que los paquetes dirigidos a nodos legítimos se envíen a otra dirección de enlace. Esto se hace enviando mensajes NS o NA con la opción de Source Link-Layer Address o Target Link-Layer Address respectivamente modificada.
  - Este tipo de ataque crea entradas falsas en la cache de vecinos de la víctima.
  - La víctima podría enviar información al nodo malicioso, abriendo la posibilidad de un ataque de Man in the Middle o un DoS.

- **Neighbor Unreachability Detection (NUD) Failure**:
  - Cuando no se ha efectuado comunicación entre un par de nodos por un largo período de tiempo, es posible que se utilice el algoritmo NUD para comprobar la permanencia del nodo en el enlace.
  - Si no hay respuesta luego de varias solicitudes, la entrada de la cache de vecinos asociada al nodo es borrada.
  - El atacante puede forjar mensajes NA falsos en respuesta a los mensajes NS durante el proceso de NUD, provocando una entrada en la cache de vecino de la víctima.
  - Este tipo de ataque puede convertirse en un DoS si el atacante mantiene esta técnica durante un largo período de tiempo.

- **Duplicate Address Detection DoS attack**:
  - El atacante envía mensajes NA cuando un nuevo host intenta verificar la unicidad de una dirección (DAD) para autoconfigurarse.
  - La víctima no logra configurar una dirección IPv6, por lo que se mantiene ejecutando continuamente el algoritmo DAD.
  - Con este ataque se puede impedir la comunicación de una gran cantidad de hosts al mismo tiempo.

- **Malicious Last Hop Router**:
  - Un router malicioso puede enviar mensajes RA modificados en respuesta a un mensaje NS.
  - La víctima podría configurar el router malicioso como su router por defecto.
  - El atacante podría dirigir los paquetes al nodo malicioso comprometiendo los datos.

- **Spoofed Redirect Message**:
  - El atacante forja un mensaje Redirect falso con la dirección del router por defecto de la víctima.
  - El atacante hace que la víctima envíe los paquetes al nodo malicioso.

- **Bogus On-Link Prefix**:
  - El atacante envía mensajes RA indicando que un prefijo de una cierta longitud se encuentra dentro del enlace.
  - Los nodos que reciben dicho mensaje no enviarán al router por defecto los paquetes dirigidos a las direcciones que se encuentren dentro de los prefijos previamente anunciados.
  - Las víctimas enviarán mensajes NS. Al no haber respuesta se produce un ataque DoS.
  - El atacante puede controlar el tiempo del ataque con el "Lifetime" asociado al anuncio del prefijo.
  - Si el atacante envía un mensaje en respuesta a la solicitud, podría convertirse en un ataque de Man in the Middle.

- **Bogus Address Configuration Prefix**:
  - El atacante envía mensajes RA anunciando un prefijo falso para la subred.
  - Los nodos que reciben dicho mensaje se autoconfiguran con una dirección no válida.
  - Como resultado, los mensajes de respuesta nunca serán recibidos ya que la dirección fuente del hosts no es válida.

- **Parameter Spoofing**:
  - Los mensajes RA contienen parámetros que ayudan a la autoconfiguración de un host.
  - Si dichos parámetros se han falsificado, los nodos pueden ser forzados a seguir reglas que impidan que los hosts se comuniquen con otros nodos.
  - Por ejemplo, si el valor del campo *Current Hop Limit* (ver Figura 2.6) es configurado con un número pequeño, los paquetes pueden ser desechados antes de llegar a su destino.

- **Replay Attacks**:
  - El atacante puede configurarse como un host válido, mediante la captura de tráfico realizado por el intercambio inicial de un nodo que se une al enlace.
  - Cuando el nodo real queda fuera de servicio, el atacante puede tomar su lugar y establecer la comunicación con otros, como si fuese el nodo original.

- **Neighbor Discovery DoS Attack**:
  - El atacante puede mantener el router local ocupado con el envío constante y masivo de mensajes ND.
  - Todas las solicitudes hechas por los nodos legítimos serán desechadas por completo, dado que el router está procesando otras solicitudes maliciosas.

2.2 Secure Neighbor Discovery

El protocolo ND es el responsable de encontrar otros hosts dentro del mismo segmento de red. Sus especificaciones [1] dejan abierta la posibilidad de ejecutar varios ataques y por tal motivo sugiere tener en cuenta ciertas consideraciones de seguridad, dentro de las cuales se encuentra el uso de IPsec y SEND como mecanismos de resguardo ante los ataques ya mencionados.

Una de las claves del protocolo ND es dejar que el proceso de autoconfiguración de direcciones ocurra de manera automática, sin intervención humana. Con IPsec el intercambio de claves se puede dar de manera automática siempre y cuando los hosts ya tengan una dirección IPv6 establecida; en caso contrario, cuando un host no tiene aún una dirección válida, IPsec no es capaz de realizar el intercambio automático de claves, por lo cual su uso se torna tedioso e impracticable.

El protocolo SEND (Secure Neighbor Discovery) [8] está diseñado para reducir la exposición a los ataques y amenazas en las que se ve envuelto ND. Para ello introduce nuevos mensajes y opciones dentro de las ya existentes en el protocolo ND, así como mecanismos que permiten defenderse de los ataques basados en identidad e integridad.

Los principales componentes usados por el protocolo SEND son los siguientes [9]:

- ***Certification Paths***: basados en anclas de confianza que certifican la autoridad de los routers para actuar como tal y que le permite a los hosts ser configurados con dichas anclas antes de adoptar a su router por defecto. Se introducen dos nuevos mensajes ICMPv6 para la obtención de dichos caminos: Certification Path Solicitation (CPS) y Certification Path Advertisement (CPA).
- ***Cryptographically Generated Addresses (CGA)***: son usadas para asegurar que el remitente del mensaje es, en efecto, el propietario de la dirección reclamada. Esto se logra gracias a la capacidad de utilizar una clave pública como dirección IPv6 en todos los nodos. La nueva opción creada para este fin recibe el nombre de CGA.
- ***RSA Signature***: es también una nueva opción, la cual permitirá proteger la integridad de los mensajes, así como la autenticación de la identidad del remitente. Se usa en conjunto con la opción CGA.
- ***Timestamp y Nonce***: estas dos nuevas opciones permiten prevenir ataques de reproducción.

2.2.1 Mensajes

Se definen dos nuevos mensajes ICMPv6 (ver Tabla 2.3) para ser usados en el proceso de ADD (Autorization Delegation Discovery) que luego permitirán verificar la identidad de un host.

ICMPv6 Type	ND Message
148	Certification Path Solicitation (CPS)
149	Certification Path Advertisement (CPA)

Tabla 2.3: ICMPv6 Type para mensajes SEND

2.2.1.1 Certification Path Solicitation (CPS)

Este mensaje es enviado por los nodos durante el proceso de ADD para obtener como respuesta, un mensaje CPA emitido por un router del enlace. La Figura 2.15 (tomada de [8]) muestra la estructura del mensaje CPS.

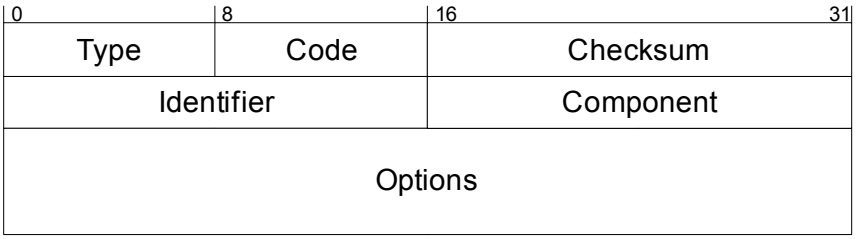

Figura 2.15: Estructura CPS

Los campos de la cabecera IPv6 deben contener la siguiente información:

Source Address: si la interfaz no ha sido configurada se utiliza la dirección IPv6 no especificada (::), de lo contrario, se usa la dirección IPv6 unicast de tipo Link-Local.

Destination Address: puede ser la dirección IPv6 multicast de All-Routers, Solicited-Node o la dirección del router por defecto del host.

Hop Limit: el valor debe ser 255.

A nivel de ICMPv6, los campos deben contener la siguiente información:

Type: debe ser 148.

Code: debe ser 0.

Identifier: el valor es un número entero generado de forma aleatoria y diferente de cero, usado para evitar colisiones y debe coincidir con el respectivo campo *Identifier* del mensaje CPA (ver Figura 2.16).

Component: el valor es un número entero que puede ser 65.535 si el remitente quiere recuperar todos los certificados ó el valor del identificador del componente correspondiente al certificado que el receptor quiere recuperar.

Option: debe usar la opción Trust Anchor, en este caso puede haber una o varias anclas de confianza que el cliente está dispuesto a aceptar.

2.2.1.2 Certification Path Advertisement (CPA)

Los mensajes CPA se envían en respuesta a los mensajes CPS. Los routers que emiten estos mensajes están autorizados por un ancla de confianza. La Figura 2.16 (tomada de [8]) muestra la estructura de este mensaje.

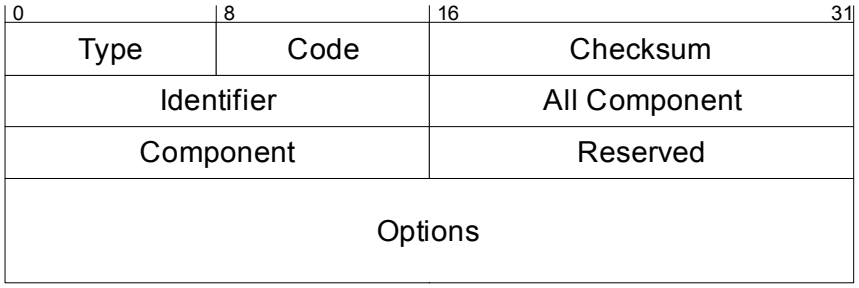

Figura 2.16: Estructura CPA

Los campos de la cabecera IPv6 deben contener la siguiente información:

Source Address: la dirección IPv6 unicast de tipo Link-Local asignada a la interfaz por donde será enviado el mensaje.

Destination Address: puede ser la dirección IPv6 multicast de All-Nodes o Solicited-Node del receptor.

Hop Limit: el valor debe ser 255.

A nivel de ICMPv6, los campos deben contener la siguiente información:

Type: debe ser 149.

Code: debe ser 0.

Identifier: el valor es un número entero diferente de cero cuando no es enviado a la dirección multicast de All-Nodes y debe coincidir con el respectivo campo *Identifier* del mensaje CPS (ver Figura 2.15) previamente solicitado.

All Component: el valor es un entero usado para informar al receptor del número de certificados en el camino. Un mensaje CPA sólo deberá enviarse por componentes separados si hay más de un certificado en el camino. Esto es para evitar la fragmentación excesiva en la capa IP.

Component: el valor es un entero diferente de cero, usado para informar al receptor cuál certificado está siendo enviado. Un valor igual a cero, indica que no hay más componentes que vienen en el anuncio.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Option: puede usar la opción Trust Anchor o Certificate, en el primer caso puede haber una o varias anclas de confianza que ayuden a los receptores a decidir qué anuncios son útiles para ellos; en el caso de usar la opción de Certificate, sólo un certificado es proporcionado en cada opción para establecer la parte de un camino de certificación a un ancla de confianza. Ambas opciones

pueden estar presentes, en este caso la opción de Trust Anchor debe aparecer en el primer componente de un anuncio multi-componente.

2.2.2 Opciones

El protocolo SEND introduce seis nuevas opciones que permiten reforzar el proceso de ND. Dichas opciones se muestran en la Tabla 2.4.

Type	Option Name
11	Cryptographically Generated Address (CGA)
12	RSA Signature
13	Timestamp
14	Nonce
15	Trust Anchor
16	Certificate

Tabla 2.4: Options Type para SEND

2.2.2.1 Cryptographically Generated Address

Son direcciones IPv6 [10] donde algunos trozos de la dirección son generados por una función hash, a partir de una clave pública y parámetros auxiliares; la correspondiente clave privada se usa para firmar los mensajes enviados.

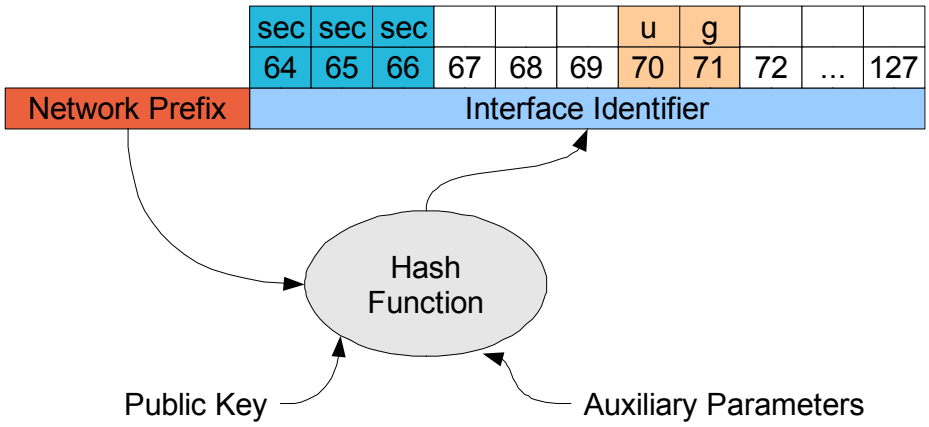

Figura 2.17: CGA

La Figura 2.17 muestra una breve descripción de cómo se forma una dirección CGA y los parámetros de entrada que se requieren.

CGA contiene un parámetro de seguridad (*Sec*) que determina su fortaleza contra ataques de fuerza bruta. Es un número que se codifica en los tres primeros bits del identificador de la interfaz (ver Figura 2.17).

En IPv6 los bits 6 y 7 del identificador de la interfaz son banderas (flags) [11] (ver Figura 2.17). El bit 6 (U/L bit) se utiliza para determinar si la dirección se administra

de forma universal o local; mientras que el bit 7 (I/G bit) se utiliza para determinar si la dirección es individual (unicast) o de grupo (multicast o broadcast).

Durante el proceso de generación de la dirección CGA son calculados dos resúmenes hash. El *Hash1* tiene un tamaño de 64 bits mientras que el *Hash2* tiene un tamaño de 112 bits, ambos cálculos usan el algoritmo SHA-1.

Cada CGA está asociado con una estructura de datos conocida como los parámetros CGA y contiene el formato mostrado en la Figura 2.18 (tomada de [9]).

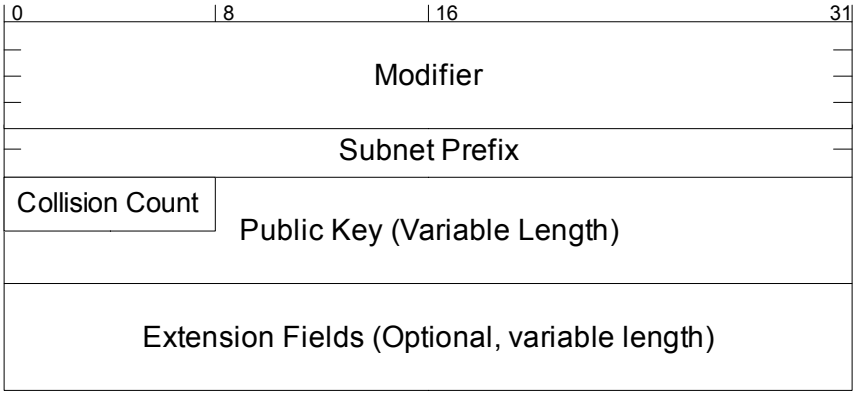

Figura 2.18: CGA Parameter Data Structure

Los campos del CGA Parameter deben contener la siguiente información:

Modifier: es un valor entero generado automáticamente.

Subnet Prefix: contiene los 64 bits del prefijo de subred CGA.

Collision Count: es un valor que se incrementa durante la generación CGA para reponerse de la colisión de una dirección detectada durante el proceso DAD.

Public Key: contiene la clave pública del propietario de la dirección. Debe estar en formato ASN.1 como una codificación DER (Distinguished Encoding Rules).

Extension Fields: es opcional y actualmente no es usado.

Como se describió anteriormente, dos resúmenes hash son calculados usando el algoritmo SHA-1 y la entrada son los parámetros CGA. Para el *Hash1*, los 64 bits finales se obtienen de los 64 bits más a la izquierda del resumen de 160 bits que arroja SHA-1; mientras que para *Hash2*, se toman los 112 bits más a la izquierda del resumen de 160 bits de SHA-1; pero, la entrada en este caso varía, ya que los campos *Subnet Prefix* y *Collision Count* son fijados en cero.

El proceso para la generación y verificación es el siguiente:

A. Generación de la CGA

El proceso de generación toma tres entradas, 64 bits del prefijo de subred, la clave pública y el parámetro de seguridad (*Sec*). El costo de generar la CGA depende, exponencialmente, del parámetro de seguridad.

1. Se genera un valor aleatorio de 128 bits llamado modificador (Modifier).
2. Se concatena el modificador junto con nueve bytes fijados a cero a la clave pública previamente codificada y se ejecuta el algoritmo SHA-1. Se toman los 112 bits más a la izquierda, en este caso se obtiene el *Hash2*.
3. Se hace un AND lógico entre una máscara (16**Sec*) y el *Hash2*. Si el resultado es cero se continúa el proceso, si no se incrementa en uno el modificador y regresa al paso 2.
4. Se fijan los 8 bits del campo Collision Count a cero.
5. Se concatena el prefijo de subred, el Collision Count y la clave pública codificada al modificador final. Se ejecuta el algoritmo SHA-1 y se toman los 64 bits más a la izquierda, en este caso se obtiene el *Hash1*.
6. Se forma un identificador de interfaz a partir del *Hash1*, haciendo los cambios en los bits correspondientes al Sec, "u" y "g". Los bits "u" y "g" representan los bits 6 y 7 y deben ser fijados en cero [11].
7. Se concatenan los 64 bits del prefijo de subred con los 64 bits del paso 6 para formar la dirección IPv6.
8. Se ejecuta el algoritmo DAD si es necesario. Si es detectada una colisión, se debe incrementar el valor del campo Collision Count en uno (1) y regresar al paso 5. Luego de tres colisiones se debe abortar el proceso y reportar un error.
9. Por último se forma la estructura de parámetros CGA con los valores finales de cada campo, obtenidos durante el proceso de generación

B. Verificación de la CGA

El proceso de verificación de la CGA toma dos entradas, la dirección (CGA) y la estructura de parámetros CGA.

1. Se verifica que el valor del campo Collision Count sea 0, 1 o 2, si no la verificación falla.
2. Se verifica que el campo Subnet Prefix sea igual al prefijo de subred de la dirección (CGA) del paquete entrante, si difieren la verificación falla.
3. Se ejecuta el algoritmo SHA-1 para obtener el *Hash1*.
4. Se compara el valor del paso 3 con los 64 bits del identificador de la interfaz de la dirección (CGA), la diferencia en los bits *Sec*, "u" o "g" son ignorados. Si difiere el resultado, la verificación falla.
5. Se toman los bits correspondientes al parámetro de seguridad (*Sec*).
6. Se concatena el modificador con nueve bytes fijados en cero con la clave pública y se ejecuta el algoritmo SHA-1. Se obtiene el *Hash2*.
7. Se compara el resultado de un AND lógico entre la máscara (16**Sec*) y el *Hash2*, si el resultado es diferente de cero la verificación falla.

Si la verificación fue exitosa, el verificador tiene la certeza de que la clave pública realmente pertenece a la dirección IPv6 del paquete entrante.

Si la verificación falla en alguno de los pasos, se debe detener la ejecución del proceso de verificación.

SEND usa CGA en el intercambio de los mensajes ND como mecanismo de prueba de autenticidad del mensaje, sin necesidad de establecer un sistema de distribución de claves públicas (PKI). La estructura de la opción CGA se muestra en la Figura 2.19 (tomada de [8]).

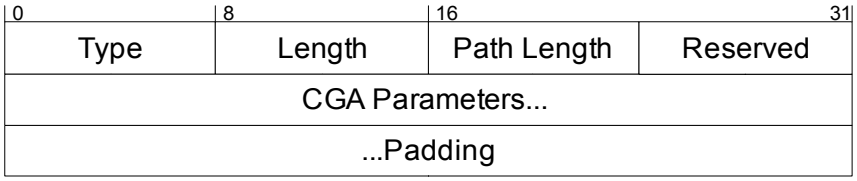

Figura 2.19: Estructura CGA Option

Los campos de la opción CGA deben contener la siguiente información:

Type: debe ser 11.

Length: es el valor de la longitud de la opción. Deben incluirse todos los campos para el cálculo.

Pad Length: corresponde al número de bytes de relleno a usar en el campo *Padding*.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

CGA Parameter: contiene la estructura de parámetros CGA definida previamente.

Padding*:* el valor debe ser cero y debe contener la cantidad de bytes especificados en el campo *Pad Length*.

Si un nodo está configurado para usar la opción CGA, entonces está opción debe estar presente en los mensajes NS, NA, RS, RA y Redirect.

Dependiendo del tipo de mensaje la dirección (CGA) puede aparecer en diferentes sitios:

- En los mensajes NS, la dirección deberá ser la especificada en el campo *Target Address* para las solicitudes que generan un proceso DAD, de otro modo deberá ser la dirección origen del mensaje.
- En los mensajes NA, RS, RA y Redirect, la dirección deberá ser la dirección origen del mensaje. La opción CGA no es usada en los mensajes RS cuando la dirección de origen es la no especificada (::).

Los nodos configurados para utilizar la opción CGA, deben realizar la verificación CGA en conformidad con lo antes descrito y la longitud mínima para el manejo de las claves debe ser al menos de 1.024 bits y una máxima de 2.048 bits.

La Figura 2.20 muestra como estaría estructurado un paquete IPv6 que contiene la opción CGA.

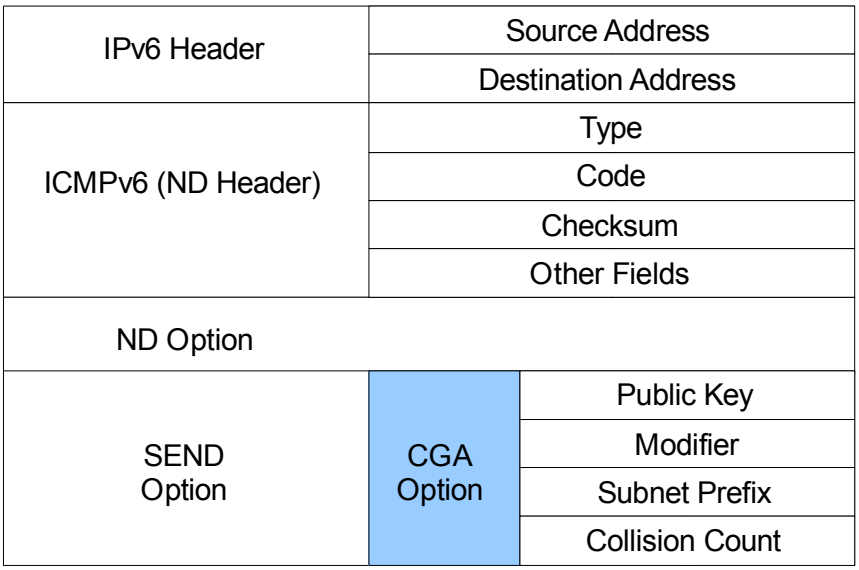

Figura 2.20: Paquete IPv6 con CGA Option

Actualmente existe una extensión al formato CGA que permite el soporte a múltiples resúmenes hash [12].

2.2.2.2 RSA Signature

Esta opción permite agregar una firma digital a los mensajes ND para poder establecer la autenticidad del los paquetes intercambiados. La estructura de esta opción se muestra en la Figura 2.21 (tomada de [8]).

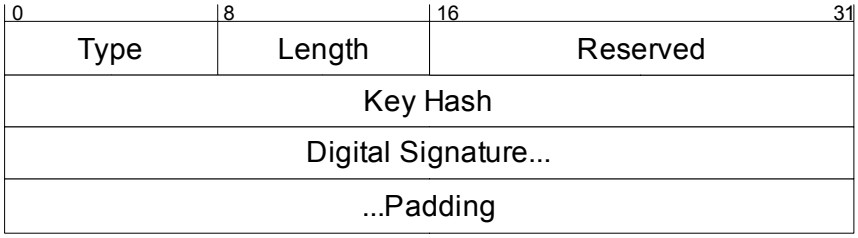

Figura 2.21: Estructura RSA Option

Los campos de la opción RSA Signature deben contener la siguiente información:

Type: debe ser 12.

Length: el valor de la longitud de la opción. Deben incluirse todos los campos para el cálculo.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Key Hash: contiene los 128 bits más a la izquierda del resumen que se obtiene al computar el algoritmo SHA-1 a la clave pública usada para construir la firma,

la cual es tomada del campo *Public Key* del parámetro CGA llevada en la opción CGA.

Digital Signature: este campo es de longitud variable y contiene una firma RSA de tipo PKCS#1 v1.5 construida por el remitente del mensaje con su clave privada. La firma es calculada con el algoritmo RSASSA-PKCS1-v1.5 y SHA-1, a partir de la siguiente secuencia de bytes:

1. 128 bits que representa una etiqueta creada por el editor de la especificación SEND, (Tag: 0x086F CA5E 10B2 00C9 9C8C E001 6427 7C08).
2. 128 bits de la dirección de origen tomada de la cabecera IPv6.
3. 128 bits de la dirección de destino tomada de la cabecera IPv6.
4. 32 bits (8 bits del campo Type, 8 bits del campo Code, 16 bits del campo Checksum) de la cabecera ICMP.
5. La cabecera ND comenzando desde el byte siguiente al campo *Checksum* de la cabecera ICMPv6 hasta el comienzo de la opción ND.
6. Todas las opciones ND que preceden la opción RSA Signature.

Padding: contiene los bits de relleno, deben estar en cero.

Si un nodo está configurado para usar esta opción, entonces debe estar presente en los mensajes NS, NA, RS, RA y Redirect. Esta opción no es usada en los mensajes RS cuando la dirección de origen es la no especificada.

Todos los nodos que estén configurados para utilizar esta opción deben permitir los diferentes métodos de autorización del remitente para cada tipo de mensaje Neighbor Discovery por separado. La autoridad del remitente puede ser verificada por medio de un ancla de confianza, CGA o inclusive ambos.

Todos los nodos deben llevar un registro que contenga la información siguiente:

- Una pareja de clave (Keypair). Si una autoridad de delegación está siendo usada, debe existir el camino de certificación desde un ancla de confianza al par de claves.
- Una bandera CGA (CGA flag) que indique si se está usando o no CGA.

La Figura 2.22 representa un esquema visual de los datos que contiene la opción RSA Signature.

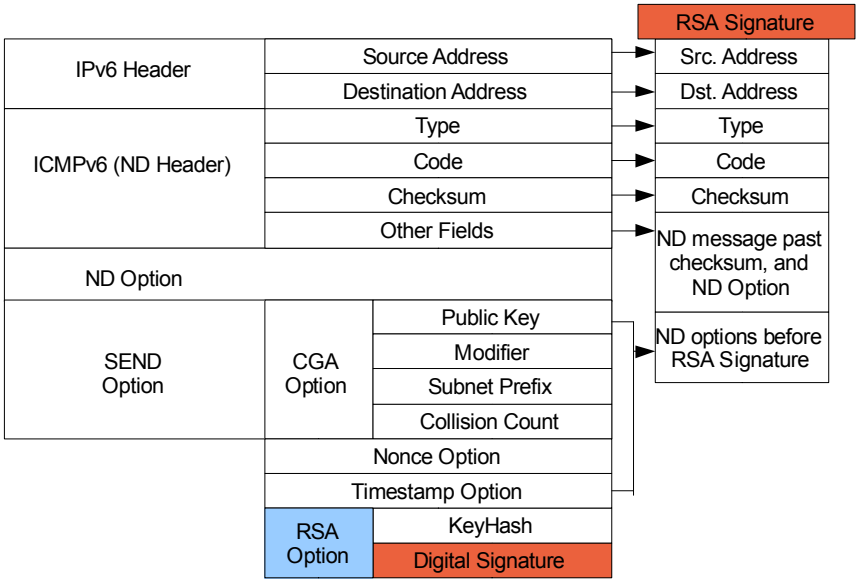

Figura 2.22: RSA Digital Signature contents

El receptor del mensaje debe ignorar cualquier opción que esté después de la firma RSA.

2.2.2.3 Timestamp

Esta opción tiene por finalidad asegurar que los mensajes de anuncio no solicitados y las redirecciones no sean repetidas. La Figura 2.23 (tomada de [8]) muestra la estructura de la opción Timestamp.

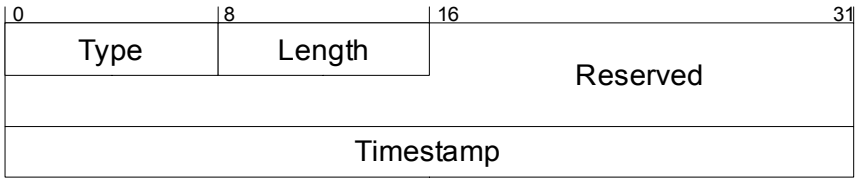

Figura 2.23: Estructura Timestamp Option

Los campos de la opción Timestamp deben contener la siguiente data:

Type: debe ser 13.

Length: el valor de la longitud de la opción. Deben incluirse todos los campos para el cálculo.

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Timestamp: representa el número de segundos transcurridos desde las 00:00:00 del 1 de Enero de 1970 GMT. Sigue el estándar en los sistemas Unix.

Los nodos deben ser configurados para manejar tres valores:

1. TIMESTAMP_DELTA (300 segundos); representa la marca de tiempo permitida.
2. TIMESTAMP_FUZZ (1 segundo); representa la marca de tiempo para comparaciones.
3. TIMESTAMP_DRIFT (1 %, 0.01); representa la marca de tiempo para el movimiento de reloj permitido.

Para facilitar la validación de las marcas de tiempo, cada nodo debe almacenar el tiempo en que se recibió y aceptó el último mensaje SEND. Este tiempo recibe el nombre de RDlast, mientras que la marca de tiempo del último mensaje recibido y aceptado se recibe el nombre de TSlast.

La verificación del Timestamp por los receptores se realiza de la siguiente forma:

- Cuando el mensaje recibido proviene de un nuevo nodo (un nodo no almacenado en cache), el Timestamp recibido (TSnew) es comprobado y aceptado si dicho valor es bastante reciente en comparación al tiempo de recepción del mismo (RDnew).

  -Delta < (RDnew - TSnew) < +Delta.

  El RDnew y el TSnew pasan a la cache como RDlast y TSlast.
- Si el Timestamp no está dentro de los límites, pero el mensaje es un NS al que el receptor debería responder, este debe hacerlo; pero no se debe crear una entrada en la cache. Esto permite que los nodos que tienen grandes diferencias de tiempo en sus relojes, puedan seguir comunicándose el uno con el otro.
- Cuando el mensaje recibido proviene de un nodo conocido (ya existe una entrada en la cache), el Timestamp es verificado contra el recibido previamente.

  TSnew + fuzz > TSlast + (RDnew - RDlast) x (1 - drift) - fuzz

Si la desigualdad no se mantiene el receptor deberá desechar el mensaje, en caso contrario deberá procesar el mensaje. Si TSnew > TSlast, el receptor debe actualizar el RDlast y el TSlast.

Los mensajes recibidos con la opción RSA Signature y sin la opción de Timestamp deben ser desechados.

2.2.2.4 Nonce

Esta opción se usa para asegurarse de que un anuncio es una nueva respuesta frente a una solicitud previamente hecha por un nodo. La estructura de esta opción se muestra en la Figura 2.24 (tomada de [8]).

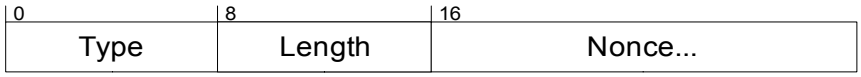

Figura 2.24: Estructura Nonce Option

Los campos de la opción Nonce deben contener la siguiente información:

Type: debe ser 14.

Length: el valor de la longitud de la opción. Deben incluirse todos los campos para el cálculo.

Nonce: el valor es un número aleatorio seleccionado por el remitente del mensaje; la longitud debe ser al menos de 6 bytes.

Todos los mensajes de solicitud y anuncios enviados por los nodos deben contener esta opción; los nodos que envían mensajes de solicitud deben almacenar internamente el valor enviado, para poder así reconocer la respuesta, la cual debe contener una copia del Nonce.

Cabe destacar que un router puede enviar una respuesta a un host específico o puede enviar un mensaje multicast a todos los nodos; el router opcionalmente puede incluir el valor *Nonce* del host que provocó el anuncio multicast, en el caso de que este valor sea omitido, puede hacer que el host no acate el anuncio, a menos que los relojes estén perfectamente sincronizados de modo que la opción *Timestamp* es requerida en todos los mensajes.

Por otro lado, todos los mensajes de solicitud recibidos con la opción RSA Signature y sin la opción Nonce, deben ser desechados. Todos los anuncios enviados a una dirección unicast con la opción RSA Signature que no contenga la opción Nonce, deben ser procesados como mensajes de anuncios no solicitados.

Aquellos anuncios que contenga la opción Nonce y no coincida con el valor *Nonce* enviado en la solicitud, deben ser desechados automáticamente.

2.2.2.5 Trust Anchor

Esta opción sólo es usada en los mensajes de camino de certificación, tanto en los anuncios como en las solicitudes (CPA y CPS). La estructura de esta opción se muestra en la Figura 2.25 (tomada de [8]).

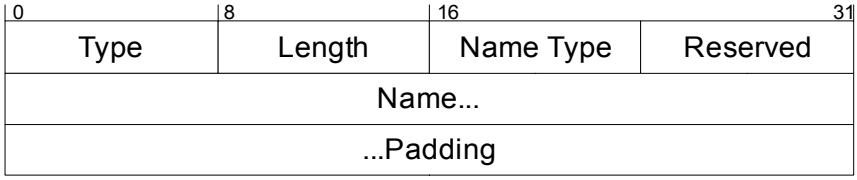

Figura 2.25: Estructura Trust Anchor Option

Los campos de la opción Trust Anchor deben contener la siguiente información:

Type: debe ser 15.

Length: el valor de la longitud de la opción. Deben incluirse todos los campos para el cálculo.

Name Type: el tipo de nombre incluido en el campo Name, puede ser:

- DER Encoded X.501 Name.
- FQDN (Fully Qualified Domain Name).

Pad Length: corresponde al número de bytes de relleno a usar en el campo Padding.

Name: Cuando el valor del campo *Name Type* es fijado a 1, el campo *Name* contiene el valor codificado en ASN.1 para el certificado X.509. Si el valor del *Type Name* es 2, el campo *Name* contiene sólo caracteres ASCII.

Padding: el valor debe ser cero y debe contener la cantidad de bytes especificados en el campo *Pad Length.*

2.2.2.6 Certificate

Esta opción sólo se usa en los mensajes CPA. La Figura 2.26 (tomada de [8]) muestra la estructura de esta opción.

Figura 2.26: Estructura Certificate Option

Los campos de la opción Trust Anchor deben contener la siguiente información:

Type: debe ser 16.

Length: el valor de la longitud de la opción. Deben incluirse todos los campos para el cálculo.

Cert Type: especifica el tipo de certificado incluido en el campo *Certificate.*

Reserved: debe ser inicializado en cero por el remitente e ignorado por el receptor.

Certificate: cuando el campo *Cert Type* es fijado a 1, el campo *Certificate* contiene un certificado de tipo X.509v3.

Padding: contiene los bits de rellenos, todos fijados a cero.

2.2.3 Authorization Delegation Discovery (ADD)

SEND está basado en un modelo de confianza en el cual los nodos (routers y hosts) son configurados para actuar como tal, es decir, existe una autoridad que certifica a los routers, para así permitir a los hosts autoconfigurarse y luego seleccionar su router por defecto [8].

Para descubrir y establecer el mecanismo de confianza, SEND utiliza los dos nuevos mensajes ICMPv6 antes descritos (CPS y CPA).

Cuando un nuevo nodo se une al enlace no puede establecer una comunicación hacia afuera del mismo, esto es debido a que se debe establecer primero un camino de certificación que le permita validar los routers que debe atravesar.

Un camino de certificación es una lista de certificados de clave pública que le permite a un nodo determinado obtener la clave pública del otro. Para poder confiar en la clave obtenida, el nodo debe verificar la firma y el estado de la validez de cada certificado en la cadena, la cual se inicia desde un ancla de confianza del verificador hasta la clave pública requerida para validar el certificado del otro (ver Figura 2.27).

Figura 2.27: Certification Path

2.2.3.1 Modelo de Autorización

Durante el proceso ND un nodo puede configurarse automáticamente con la información que obtiene al momento de unirse al enlace (zero-configuration), por lo que es particularmente fácil crear routers falsos, y particularmente difícil para un nodo distinguir, entre los routers válidos y los no válidos.

El uso de las opciones CGA y RSA Signature dentro de los mensajes ND provee autenticación, pero no evita el hecho de que existan nodos maliciosos que generen sus propias claves públicas y privadas, respectivamente.

Un modelo de autorización es requerido para diferenciar entre nodos que actúan como clientes y nodos que pueden servir a otros nodos. SEND utiliza un esquema que diferencia sólo entre routers y hosts.

Los routers son autorizados por un ancla de confianza (Trust Anchor), mientras que los hosts son configurados con dichas anclas para luego validar si un router es quien dice ser. Aquí es donde SEND difiere de zero-configuration, ya que requiere que exista previamente una infraestructura que asigne los derechos o roles a los routers.

2.2.3.2 Modelo de Despliegue

SEND puede ser configurando con un modelo de despliegue con anclas de confianza de manera centralizada o descentralizada.

Un modelo centralizado asume el uso de una raíz global capaz de autorizar routers y opcionalmente el espacio de direcciones que deben anunciar, y los hosts deben ser configurados con las claves públicas de dicha raíz (ver Figura 2.28). Actualmente no existe una raíz global que permita esta configuración.

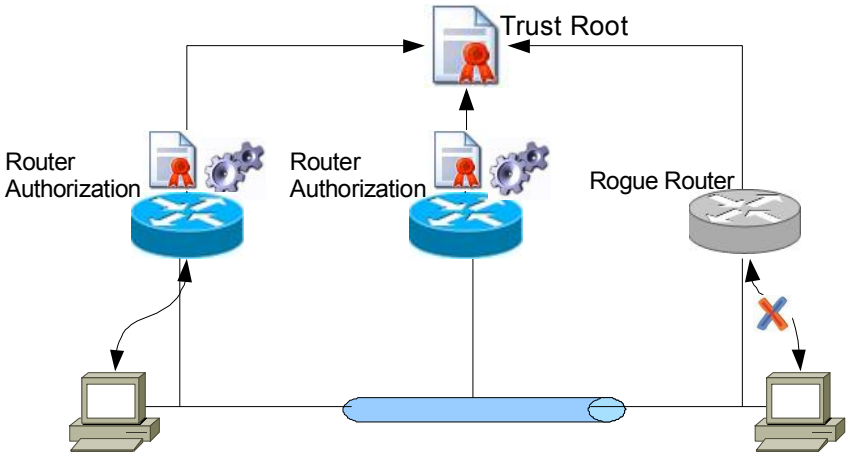

Figura 2.28: Delegate Authority Model

En un modelo descentralizado, los hosts son configurados con una colección de claves públicas, las cuales pueden ser obtenidas a través de diferentes formas (una clave pública de la organización, ISP, u otros hosts). Este modelo brinda la flexibilidad de trabajar en enlaces donde hay routers certificados o no certificados.

2.2.3.3 Formato del Certificado

SEND establece que un ancla de confianza debe certificar a los routers del enlace, para ello se utiliza un Certificado de Autorización del Router (Router Authorization Certificate) de tipo X.509v3 [13], también especifica que debe contener al menos una extensión para direcciones IPv6.

Si un camino de certificación es creado durante el proceso de descubrimiento de la autoridad, los certificados en el camino deberán contener una o varias extensiones IPv6 del certificado padre.

Un certificado X.509v3 asocia un DN (Distinguished Name) con una clave pública, es decir, consiste de un conjunto nombre-valor que identifica unívocamente a un

sujeto (en este caso un nodo SEND). Básicamente estos certificados constan de dos partes, una sección de datos y una sección de firmas; el certificado que recibe el router debe ser firmado por un ancla de confianza.

Como una extensión de direcciones IPv6 es incluida en el certificado, estas deberían tener al menos un elemento AddressesOrRange, el cual a su vez puede incluir el elemento addressPrefix o addressRange, que contendrán el prefijo de la dirección IPv6 al cual el router está autorizado a enrutar, ó la variedad de prefijos en el caso de addressRange. Si se permite anunciar cualquier prefijo, el prefijo usado será el prefijo nulo (::/0).

Si un nodo recibe un Router Authorization Certificate, debe comprobar primero si la firma del certificado fue generada por las autoridades que lo delegan. Luego deberá comprobar si todo el addressPrefix o las entradas addressRange en el certificado del router están contenidos dentro de la variedad de direcciones en el certificado.

Si un addressPrefix o addressRange no está contenido dentro de los prefijos de subred o rangos que las autoridades especifican, el host puede intentar tomar una intersección de los prefijos de subred/rango y usar esa intersección.

Si la intersección resultante es vacía, el cliente no debe aceptar el certificado. Si el addressPrefix en el certificado falta o es el prefijo nulo (::/0), el prefijo padre o el rango deberán ser usados.

Si no hay ningún prefijo padre o rango, se dice que los prefijos de subred que el router anuncia son libres, es decir, se permite que el router anuncie cualquier prefijo.

Por otra parte, los hosts deben comprobar el campo de subjectPublicKeyInfo dentro del último certificado en el camino de certificación y así asegurar que sólo claves públicas RSA son usadas para intentar la validación de firmas del router.

La Figura 2.29 (tomada de [8]) muestra un ejemplo de camino de certificación.

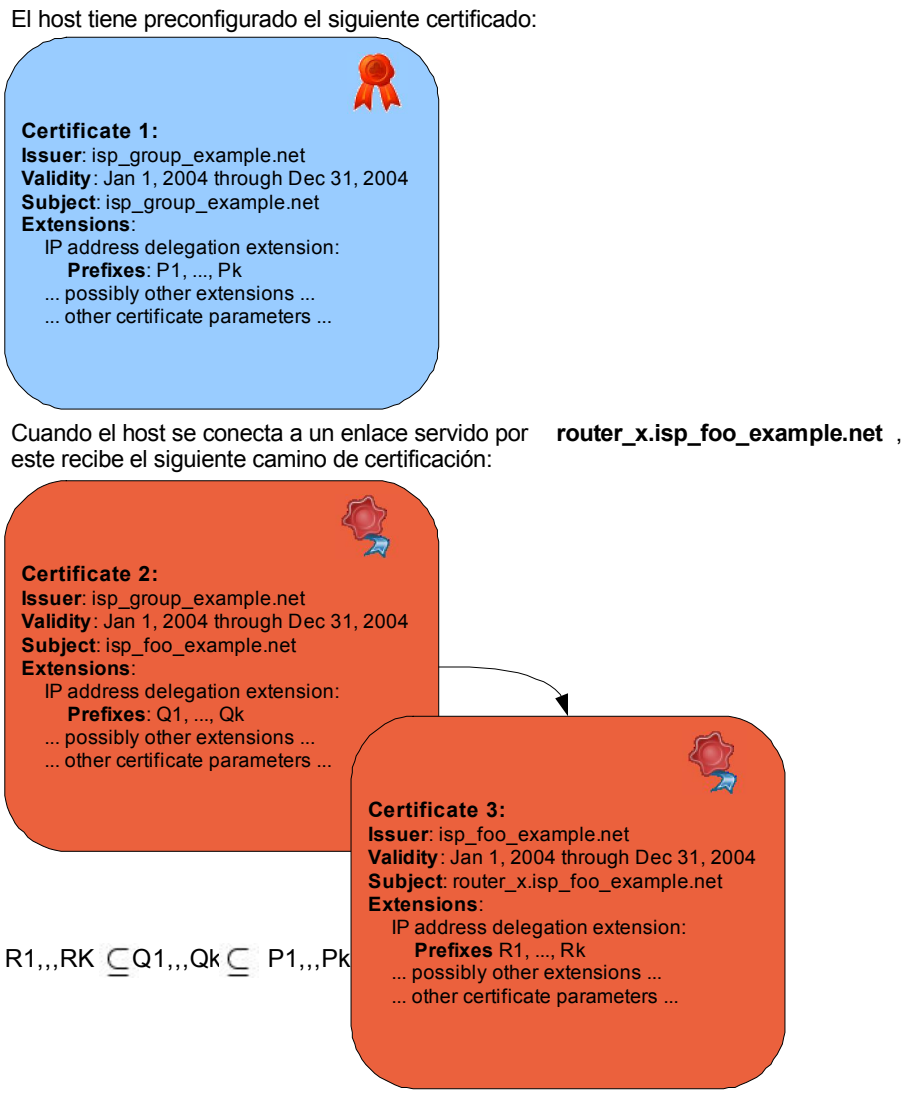

Figura 2.29: Ejemplo Camino de Certificación

Los certificados recibidos por el host deben ser comprobados, para ello es necesario contar con una conexión a Internet, por lo que no es posible realizar una Lista de Revocación de Certificados (CRL) en línea si fuese necesario; y por lo tanto la aceptación de un certificado debe considerarse provisional.

Una CRL [13] es una lista firmada digitalmente por una tercera parte de confianza (Trusted Third Party) que contiene los números seriales de los certificados revocados junto con su fecha y razón de revocación.

Como se muestra de la Figura 2.29, si un camino de certificación es válido, entonces el router *router_x.isp_foo_example.net* está autorizado a la ruta de prefijos R1,…, RK, la cual es un subconjunto de Q1,..., Qk que también a su vez es subconjunto de P1,..., Pk.

2.2.3.4 Configuración y Procesamiento de Reglas

La configuración de un ancla de confianza consiste en:

- Un algoritmo de firma de clave pública y su correspondiente clave pública.
- Un *Name* que se ubica en el campo de la opción Trust Anchor y que puede ser:
  - DER Encoded X.501 Name
  - FQDN
- Opcionalmente un identificador de clave pública.
- Opcionalmente una lista del rango de direcciones que el ancla estará autorizado a delegar.

Los hosts deben estar configurados con al menos un ancla de confianza, mientras que los routers son configurados con al menos un camino de certificación y su par de claves (pública-privada) certificadas. Las claves certificadas serán requeridas para que un camino de certificación pueda ser establecido entre el certificado del router y la clave pública del ancla de confianza.

Los routers deben enviar anuncios en respuesta a las solicitudes válidas. Si la dirección de la fuente en la solicitud es la dirección no especificada, el router debe enviar la respuesta a la dirección multicast de All-Nodes del enlace; si la dirección de la fuente es una dirección de tipo unicast, el router debe enviar la respuesta a la dirección multicast de Solicited-Node correspondiente a la dirección de la fuente.

Los routers deben enviar tantos mensajes CPA por segundos como lo indica la constante MAX_CPA_RATE (10 veces por segundo). Cuando hay demasiadas solicitudes que atender, el router deberá enviar la respuesta a la dirección de multicast de All-Nodes sin importar la dirección de la fuente que aparece en la solicitud.

Si un mensaje CPS contiene el valor 65.535 en su campo *Component*, el router deberá incluir todas las opciones de certificados convenientes de modo que el camino de certificación pueda ser establecido al ancla de confianza solicitada, de lo contrario sólo se incluirá una parte, según lo requerido en el campo *Component*.

Si el router no puede encontrar un camino al ancla solicitada, este deberá enviar un anuncio sin ningún certificado; sólo deberá incluir las opciones Trust Anchor que fueron solicitadas.

En cuanto a los hosts, estos deben almacenar los caminos de certificación obtenidos durante el descubrimiento de camino de certificación, siempre y cuando comiencen de un ancla de confianza del host y puedan ser verificados. Los routers envían los certificados uno tras otro comenzando desde el ancla de confianza.

Los mensajes CPS deben ser enviados a la dirección multicast de All-Nodes si el host no ha seleccionado un router por defecto, en caso contrario, el host deberá

enviar la solicitud a la dirección de su router por defecto. Los mensajes CPS no deberían ser enviados si el host tiene un camino de certificación actualmente válido.

Un host deberá recuperar un camino de certificación cuando un mensaje CPA enviado por un router contiene una clave pública que no está disponible en la cache de certificados del host o cuando no hay algún camino de certificación a una de las anclas de confianza del host. En este caso, el host puede enviar un mensaje CPS para recuperar el camino, si no hay respuesta en un tiempo conocido como CPS_RETRY (1 segundo), el mensaje deberá ser enviado de nuevo y, se duplica el valor por cada nueva trasmisión subsecuentemente.

Si no hay respuesta a la solicitud de recuperación del camino en un tiempo CPS_RETRAY_MAX (15 segundos), el host deberá abandonar el proceso. Si el host recibe sólo una parte del camino dentro de un tiempo CPS_RETRY_FRAGMENTS (2 segundos), se puede transmitir un mensaje CPS con el valor del campo *Component* diferente a 65.535; nuevamente, no debe excederse el tiempo CPS_RETRAY_MAX para completar la recuperación.

2.2.4 La Transición

Durante la transición para crear un entorno seguro, los nodos que estén configurados para usar SEND deben soportar el uso de los mensajes no asegurados al mismo tiempo. Un mensaje *asegurado* es aquel que contiene una opción de firma digital válida. En ambientes mixtos (SEND y non-SEND), la prioridad la tienen los mensajes asegurados.

Se puede tener también una política de configuración en la cual se ignoren los mensajes no asegurados, es decir, sería un entorno sólo asegurado por mensajes SEND.

A continuación se explica cómo trabaja SEND bajo un ambiente mixto:

- Todos los mensajes de solicitud enviados por un nodo configurado con SEND deben ser asegurados.
- Los anuncios no solicitados enviados por un nodo SEND deben ser asegurados.
- Los anuncios enviados en respuesta a una solicitud no asegurada (nodo non-SEND) deben ser asegurados, pero no deben contener la opción *Nonce*.
- Los nodos SEND que usen CGA deben ejecutar DAD. Si la dirección tentativa está en uso, este debe tratar de generar una nueva. Después de tres intentos consecutivos, se registra un error de sistema y aborta la ejecución. Cuando se genera la primera dirección tentativa durante el DAD, el nodo puede aceptar mensajes seguros o no seguros, pero a partir de la segunda, se deben ignorar los mensajes no asegurados. Un nodo puede estar configurado para ignorar los anuncios no asegurados durante la ejecución de DAD.
- Se debe tener una bandera que indique cuando un mensaje está asegurado o no para las entradas en la cache de vecinos, lista de prefijos, y lista de

router por defecto. Sólo cuando se reciben mensajes asegurados, es que se actualizan las entradas.

- Un router no certificado puede ser seleccionado por un host como su router por defecto, sólo si un router certificado no es alcanzable para un prefijo dado.

2.2.5 Consideraciones de Seguridad

SEND está diseñado para responder en gran parte a las amenazas del protocolo ND. A continuación se describe como SEND contrarresta algunos de dichos ataques:

- Ataques basados en Neighbor Unreachability Detection: SEND responde a estos ataques haciendo que el nodo que emite la respuesta al mensaje NS enviado como prueba NUD, incluya la opción de RSA Signature y una prueba de autorización para el identificador de interfaz que está usando. Si dichos requisitos no son encontrados el nodo que ejecuta el algoritmo NUD descarta las respuestas.
- Ataques basados en DAD: SEND responde estos ataques: haciendo que los mensajes NA en respuesta a los DAD incluyan la opción de RSA Signature y la prueba de autorización para usar el identificador de interfaz de la dirección que está siendo probada.
- Ataques basados en Routers Discovery: en este caso los mensajes RA deben contener la opción de RSA Signature, y dicha firma es calculada usando la clave pública del nodo para así demostrar su autorización y poder enrutar los prefijos de red contenidos en la opción de Prefix Information. El router demuestra su autorización gracias a un certificado.
- Ataques de reproducción: SEND contrarresta estos ataques con la opción *Nonce*. SEND se protege de los ataques basados en mensajes no solicitados, tal como son los mensajes RA, NA y Redirect con mensajes que incluyen la opción *Timestamp*.

SEND también podría ser vulnerable a ataques de denegación de servicio (DoS). Un atacante podría enviar una gran cantidad de caminos de certificación durante el proceso ADD. Este tipo de ataques se logra haciendo que el host gaste un tiempo considerable procesando el camino de certificación, sin embargo los hosts pueden defenderse contra estos ataques limitando la cantidad de recursos que dedican para la verificación del camino.

2.2.6 Resumen de Tablas Definidas por el Protocolo SEND

A continuación se muestran una serie de tablas que especifican los mensajes, opciones, constantes y variables definidas en el protocolo SEND.

2.2.6.1 Mensajes ND

La Tabla 2.5 muestra el valor del campo Type ICMPv6 y su mensaje ND asociado.

ICMPv6 Type	ND Message
133	Router Solicitation (RS)
134	Router Advertisement (RA)
135	Neighbor Solicitation (NS)
136	Neighbor Advertisement (NA)
137	Redirect
148	Certification Path Solicitation (CPS)
149	Certification Path Advertisement (CPA)

Tabla 2.5: Resumen mensaje ND

2.2.6.2 Opciones ND

La Tabla 2.6 muestra el valor del campo Type, junto con su respectiva opción para los mensajes ND.

Type	Option Name
1	Source Link-Layer Address
2	Target Link-Layer Address
3	Prefix Information
4	Redirected Header
5	MTU
11	Cryptographically Generated Addresses (CGA)
12	RSA Signature
13	Timestamp
14	Nonce
15	Trust Anchor
16	Certificate

Tabla 2.6: Resumen opciones ND

2.2.6.3 Constantes SEND

En la Tabla 2.7 se pueden observar las constantes definidas en SEND.

Valor	Constante
1 segundo	CPS_RETRY
2 segundos	CPS_RETRY_FRAGMENTS
15 seconds	CPS_RETRY_MAX
10 veces por segundo	MAX_CPA_RATE
0x086F-CA5E-10B2-00C9-9C8C-E001-6427-7C08	CGA Message Type name space

Tabla 2.7: Constantes SEND

2.2.6.4 Variables SEND

La Tabla 2.8 muestra las variables y los valores asociados según lo especificado en SEND.

Valor	Variable
300 segundos	TIMESTAMP_DELTA
1 segundo	TIMESTAMP_FUZZ
1 % (0.01)	TIMESTAMP_DRIFT

Tabla 2.8: Variables SEND

2.3 DoCoMo's Open Source SEND Project

2.3.1 Descripción

Open Source SEND es un proyecto que provee una implementación del protocolo SEND para IPv6; también incluye un en su distribución las librerías necesarias para generar la CGA y las extensiones de direcciones IPv6 que son incluidas dentro del certificado X.509.

La aplicación fue desarrollada por ***DoCoMo USA Labs***[4], la cual es una empresa dedicada a proveer avances en el área de las tecnologías de Internet móvil y para ello cuenta con un grupo de investigadores entre quienes se encuentra un miembro del grupo de trabajo que realizó las especificaciones de SEND [8].

Esta implementación trabaja en el espacio de usuario de Linux como un firewall que captura los paquetes entrantes y salientes durante el proceso ND. Las plataformas que actualmente soportan esta implementación son: Linux con núcleo 2.6 y FreeBSD versión 5.4.

2.3.2 Ventajas

- Provee un modo de depuración con el cual se puede visualizar a través de una shell la información relacionada con los estados internos y operaciones realizadas por la aplicación SEND.
- La aplicación contiene un par de herramientas (cgatool e ipextool) que facilita la generación de claves, direcciones CGA y la integración de la extensión IPv6 requerida en los certificados. La Figura 2.30 muestra el comando usado para generar un par de claves RSA, los parámetros CGA y la dirección CGA a partir de un valor *Sec* = 1 y un prefijo de red 2003::/64.

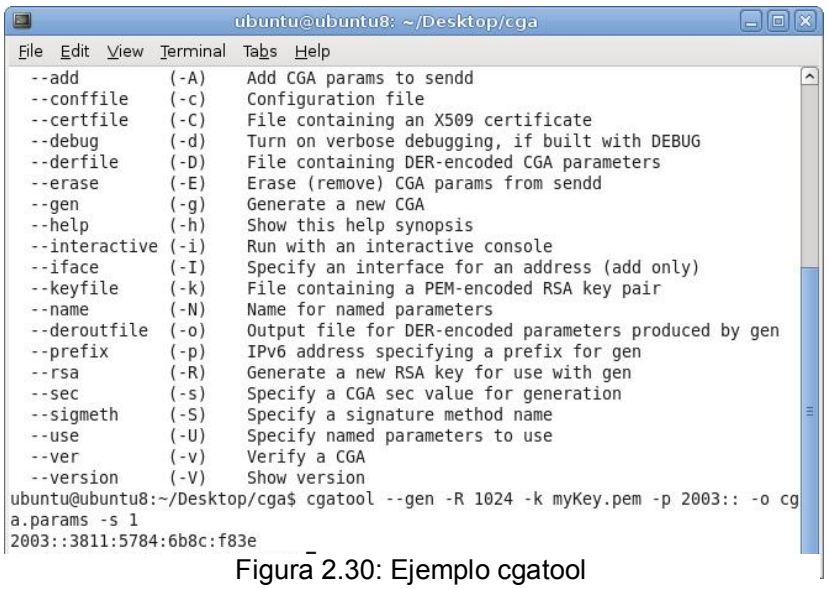

Figura 2.30: Ejemplo cgatool

[4] http://www.docomolabs-usa.com

- Provee dos tipos de configuraciones: Básica y Avanzada.
  - La configuración básica permite crear un ambiente SEND sólo para hosts, sin la presencia de routers en el enlace. La Figura 2.31 y la Figura 2.32 muestran la captura realizada durante el intercambio de mensajes NS y NA.

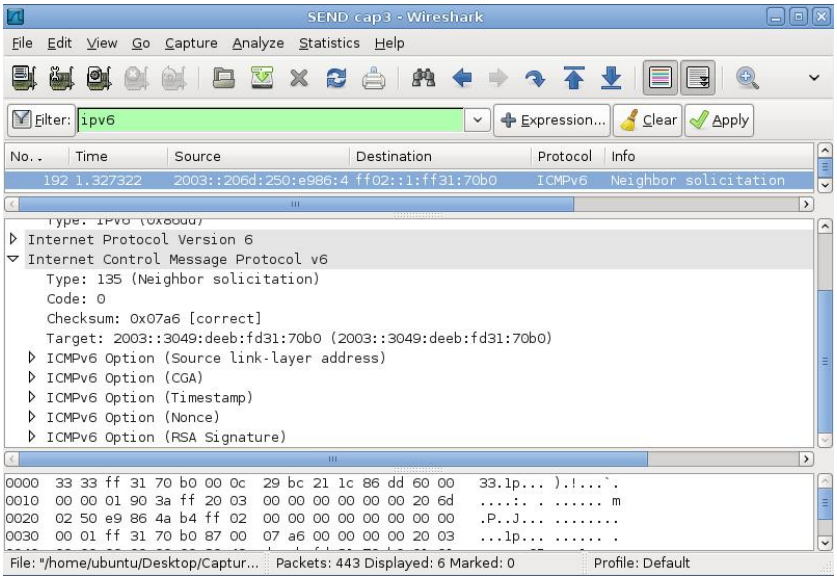

Figura 2.31: Captura de un mensaje NS con opciones SEND

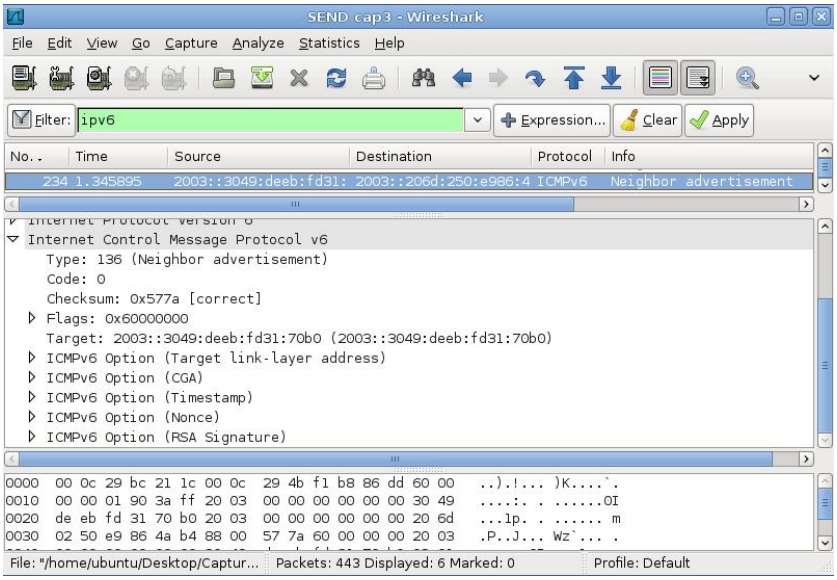

Figura 2.32: Captura de un mensaje NA con opciones SEND

- La configuración avanzada permite crear un ambiente SEND en el cual los nodos participan en el descubrimiento de los routers que residen en el enlace, usando los caminos de certificación y su correspondiente ancla de confianza. La Figura 2.33 muestra la captura de un mensaje CPA en respuesta a un CPS; mientras que la Figura 2.34 muestra la captura de un mensaje RA.

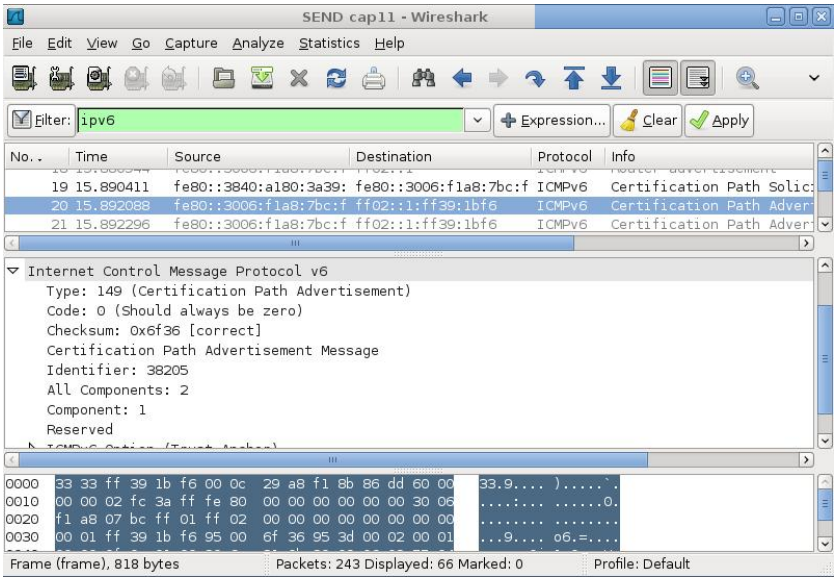

Figura 2.33: Captura de un mensaje CPA

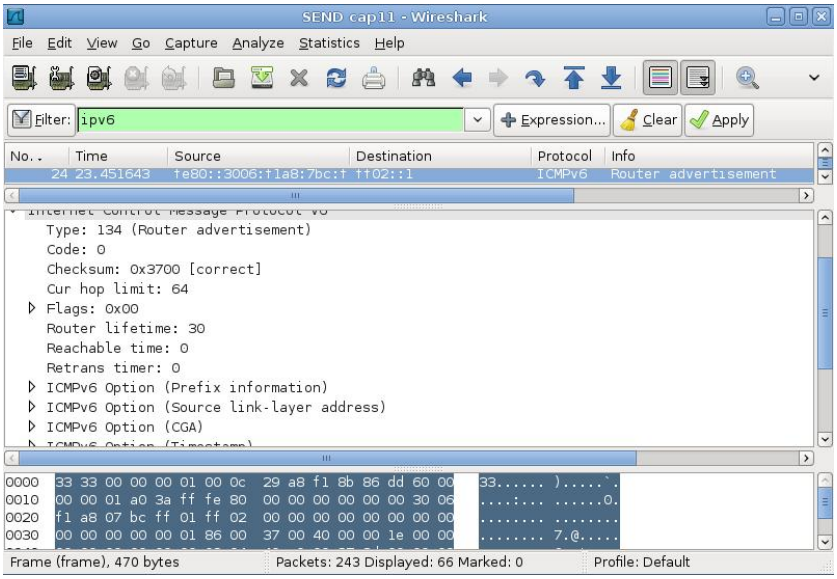

Figura 2.34: Captura de un mensaje RA que utiliza opciones SEND

2.3.3 Desventajas

- Se debe tener instalada una versión completa del sistema operativo Linux, dado que muchos de ellos actualmente no incluyen librerías de desarrollo y del núcleo necesarias para ejecutar la aplicación SEND.
- Todos los pasos para la instalación y ejecución de la aplicación SEND deben hacerse con el usuario ***root***.
- La aplicación no maneja las colisiones de dirección durante el proceso DAD, debido a la dificultad que acarrea su integración a nivel del núcleo.
- Se debe tener firme conocimiento en la configuración de las reglas del firewall (ip6tables), ya que estas son esenciales para un correcto funcionamiento de la aplicación.
- Actualmente no se brinda ningún tipo de soporte al proyecto.

3. Marco Metodológico

Para lograr los objetivos planteados previamente, es necesario definir una metodología de trabajo que permita el desarrollo de los requerimientos de la aplicación. A continuación se explican los múltiples aspectos que fueron tomados en cuenta para diseñar e implementar la aplicación.

3.1 Metodología de Desarrollo

En el proceso de desarrollo del software una de las partes más importantes es definir el esquema o enfoque a usar. La mayor parte de los modelos de proceso de software se basan en uno de los tres modelos generales: enfoque en cascada, desarrollo iterativo e ingeniería del software basada en componentes [15].

El modelo usado para la elaboración de la aplicación SEND está basado en el *enfoque en cascada* el cual consiste en cuatro fases:

- Estudio de las especificaciones (análisis de requerimientos) de la aplicación.
- Diseño.
- Desarrollo.
- Integración y pruebas.

Puesto que no existe un método ideal, a medida que se avanza por cada una de las fases del modelo de desarrollo puede ser necesario redefinir una etapa anterior.

A continuación se describen los aspectos que se tomarán en cuenta en cada una de las fases del desarrollo:

3.1.1 Análisis de los Requerimientos de la Aplicación

La aplicación cubrirá los aspectos necesarios para proporcionar un entorno seguro en un enlace IPv6, en la cual los hosts residentes no participan en el proceso de Router Discovery. Las opciones correspondientes al protocolo SEND serán agregadas a los mensajes NS y NA del protocolo Neighbor Discovery para establecer el mecanismo de confianza deseado entre los nodos. También se incluirá una herramienta que permita la generación y verificación de las direcciones CGA.

La aplicación deberá ser portable, es decir, que puede ser compilada y ejecutada en distintas plataformas Unix con un mínimo de cambios; deberá ser extensible y robusta, permitiendo incorporar módulos con nuevas funcionalidades o realizar cambios en los ya existentes. También deberá llevar un registro de trazas que permita guiar al usuario a través de mensajes.

3.1.2 Diseño de la Aplicación

Esta fase estará documentada a través de diagramas de clase utilizando notación UML.

Para esta versión de la aplicación SEND será necesario incorporar cuatro librerías: **"commons-cli-1.1.jar"**, **"log4j-1.2.15.jar"**, **"vserv-ipq-0.6.5.jar"**, **"not-yet-commons-ssl-0.3.10.jar"** y una librería dinámica llamada **"libvservipq.so"**; que ayudarán al correcto funcionamiento de la misma.

La librería **"commons-cli-1.1.jar"** forma parte de la suite de Apache Commons y provee una API para el manejo de los argumentos de entrada en línea de comando de una shell.

Con la librería de **"log4j-1.2.15.jar"** se logrará llevar el registro (logs) de los sucesos ocurridos en la aplicación y para ello cuenta con seis niveles diferentes de prioridad para los mensajes. Esta versión sólo tendrá un nivel de profundidad igual a tres que corresponden al registro de mensajes de error, debug e información. Típicamente la salida de los mensajes puede ser visualizada en tiempo de ejecución a través de la shell, pero también podrá configurase para ser redireccionada a un archivo de texto.

El diseño de la aplicación será en el espacio de usuarios (user-space) de Linux y no requerirá modificaciones a nivel del núcleo; trabajará como un firewall entre la interfaz de red y la pila IPv6, haciendo uso de las reglas de ip6tables.

Con Linux a través de las reglas de ip6tables se logra la intersección de tráfico IPv6 que entra o sale por una interfaz, es decir, que es posible manipular, autorizar o denegar el tráfico que pasa por el sistema.

Tradicionalmente se han utilizado librerías que permiten analizar el tráfico de un enlace tal y como llega a la interfaz de red, como por ejemplo **"libpcap"** o **"jpcap"**, pero hay que tener en cuenta que éste tráfico sólo se trata de una copia de lo que pasa por la red en ese instante y no se tiene la posibilidad de manipularlo, sólo se puede analizar.

La intersección de tráfico se utiliza principalmente para sistemas de seguridad e IPS (Intrusion Prevention System), la intercepción de hecho es lo que marca la diferencia entre los IPS y los IDS (Intrusion Detection System), ya que estos últimos sólo pueden detectar ataques.

En particular, los sistemas Unix siempre se han caracterizado por ser grandes sistemas orientados hacia las redes, esto hace que la gestión adecuada del tráfico de la red sea una tarea imprescindible y para ello cuenta con la librería **"libipq"** que viene en el paquete *iptables-dev*.

La librería **"libipq"** permite gestionar con total independencia en el espacio de usuario de Linux los paquetes capturados o interceptados por las reglas de ip6tables, específicamente los que son apilados por el núcleo. Por ejemplo:

```
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "133"
```

Las librería de Virtual Services IPQ **"vserv-ipq-0.6.5.jar"** y **"libvservipq.so"** son librerías que contienen o envuelven (wrapper) la librería **"libipq"**, con la cual una aplicación desarrollada en Java puede interceptar los paquetes que son encolados por las reglas de ip6tables. Eventualmente esos paquetes pueden ser manipulados o modificados para su posterior reinserción o descarte.

Por último, la librería **"not-yet-commons-ssl-0.3.10.jar"** aunque todavía no forma parte de la suite de librerías reusable de Apache Commons, es muy útil y provee los métodos necesarios para extraer la clave privada de archivos codificados con la extensión PEM (Privacy Enhanced Mail - base64) o DER (Distinguished Encoding Rules - binary format).

El diseño de la aplicación puede ser descrito a través de la arquitectura mostrada en la Figura 3.1.

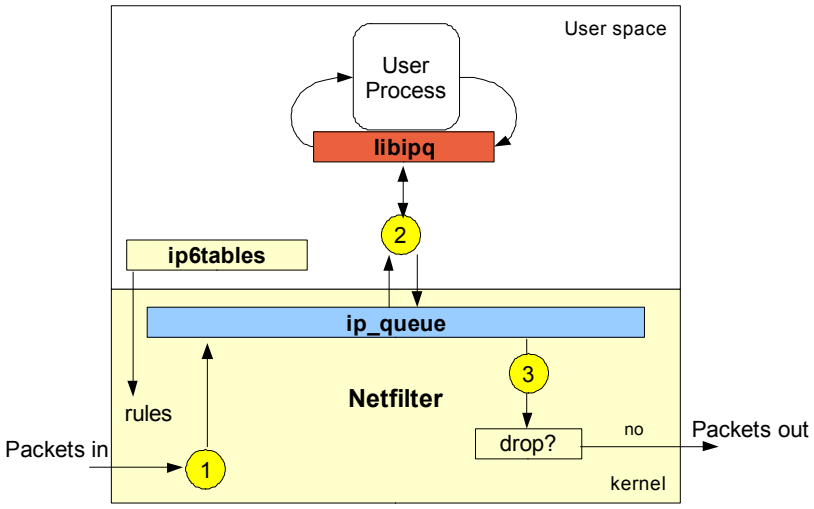

Figura 3.1: Arquitectura de la aplicación

Inicialmente serán configuradas las reglas de ip6tables que permitan encolar los mensajes ICMPv6 del protocolo ND que llegan a la interfaz de red, luego a través de la librería **"libipq"** se procederá a manipular el paquete y en base a las especificaciones del protocolo SEND se agregarán o se verificarán las opciones correspondientes al tipo de mensaje ND.

Posteriormente, el mensaje será enviado de vuelta a la cola de ip6tables y en base al análisis realizado por la aplicación dicho paquete podrá ser aceptado y enviado o rechazado y descartado.

3.1.3 Desarrollo e Implementación de la Aplicación

El desarrollo de la solución estará orientado por completo en el espacio de usuario de Linux y utilizará Java como lenguaje de programación, ya que permite la portabilidad y fácil comprensión al momento de estudiar el código escrito en cada una de las clases.

En esta fase se explicará cada uno de los recursos necesarios para la implementación de la aplicación, así como los requerimientos mínimos para su uso.

3.1.4 Integración y Pruebas

Una vez cumplido con los requerimientos mínimos de la aplicación y configuración, se procederá a realizar las pruebas para comprobar el correcto funcionamiento a través de diferentes casos de prueba.

4. Marco Aplicativo

En este capítulo se muestran los resultados obtenidos empleando la metodología descrita anteriormente (ver Capítulo 3).

4.1 Fase de Diseño

El proceso que contempla la aplicación SEND es expresado a través de las siguientes clases:

La clase ***NDMessage*** contiene los métodos necesarios para manipular un paquete IPv6 que transporta mensajes del protocolo ND. También envuelve los métodos que permiten agregar o eliminar las opciones del protocolo SEND para un paquete saliente o entrante respectivamente. Esta clase a su vez hereda de la clase ***ICMP6Packet***. La Figura 4.1 muestra el diagrama de clase asociado.

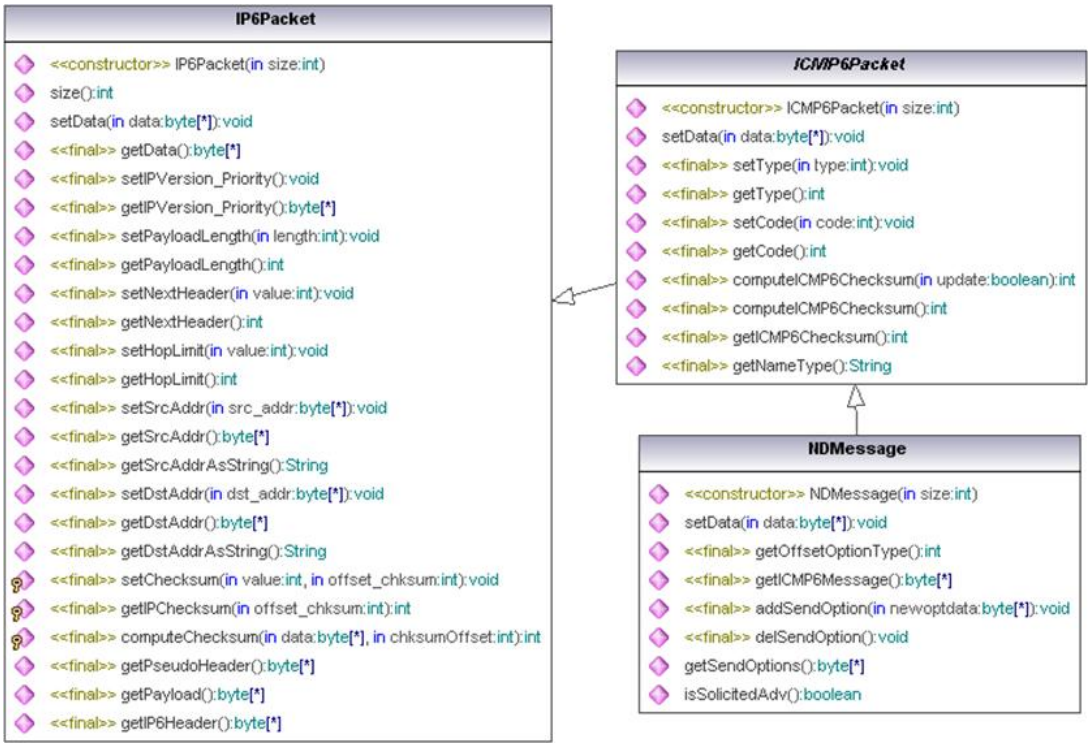

Figura 4.1: Diagrama de clase NDMessage

La clase **IP6Packet** es una clase genérica que implementa métodos que permiten extraer o agregar valores dentro de un paquete IPv6, tal como obtener la dirección fuente o dirección de destino en la cabecera (ver Figura 2.2) o cambiar el Checksum del paquete. Mientras que la clase abstracta ***ICMP6Packet*** que hereda de ***IP6Packet*** define los métodos para manipular los campos de un paquete ICMPv6.

La clase ***SENDOPT*** es una clase abstracta que define los métodos get/set de las opciones SEND y del cual heredan las siguientes clases: CGA, RSASignature, Nonce y TimeStamp como se muestra en la Figura 4.2.

Figura 4.2: Diagrama de clase SENDOPT

La clase ***CGA*** encapsula los métodos que permiten generar o verificar la opción CGA contenida en un mensaje ND, en conformidad con lo descrito en las RFCs [8] [10]. La Figura 4.3 muestra el diagrama de clase asociado.

Figura 4.3: Diagrama de clase CGA

A continuación se describe de forma visual el proceso de generación de una dirección CGA como se especifica en el Marco Teórico (ver Capítulo 2). La Figura 4.4 muestra los parámetros de entrada y el proceso de obtención del *Hash2.*

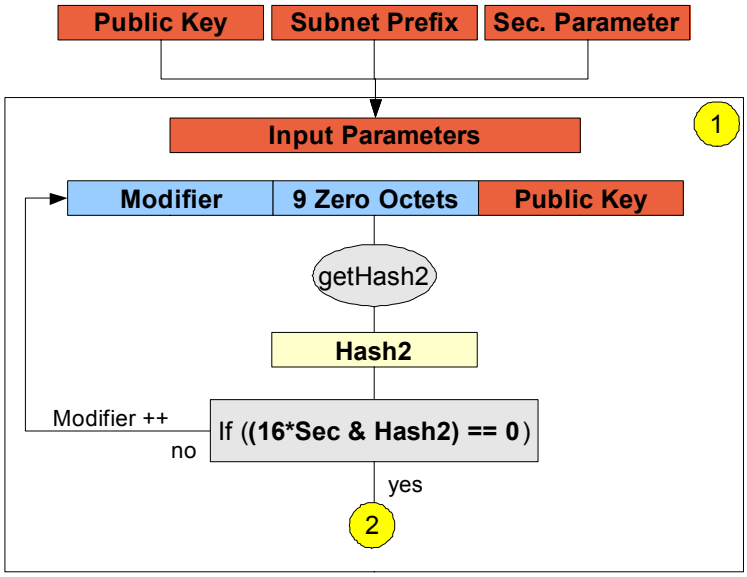

Figura 4.4: Cálculo del Hash2 en CGA

El valor del campo *Modifier* (ver Figura 2.18) es un valor aleatorio que se puede mantener constante para la generación de otras direcciones CGA con diferentes prefijos de red y así evitar la creación de una estructura de parámetros CGA por cada dirección CGA generada, reduciendo de esta forma el número de archivos de configuración asociados a un host.

La complejidad del cálculo del *Hash2* depende exponencialmente del valor del parámetro de seguridad (*Sec*). Una vez obtenido el *Hash2* se procede con el cálculo del *Hash1* como se muestra en la Figura 4.5.

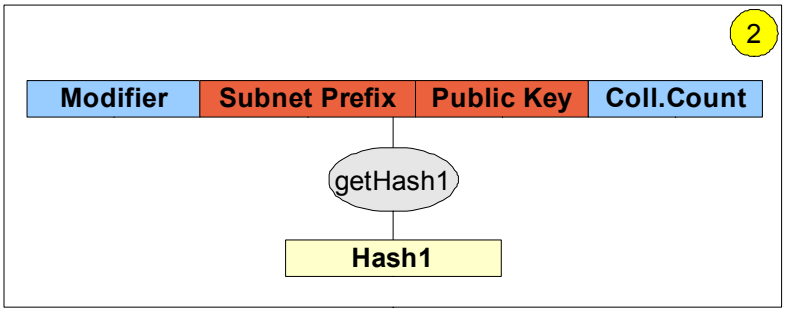

Figura 4.5: Cálculo del Hash1 en CGA

El valor inicial del campo *Collision Count* (ver Figura 2.18) es fijado en cero y su valor es incrementado en uno si se detecta duplicada la dirección CGA en el enlace. Esta versión de la aplicación no ejecutará el algoritmo DAD durante la generación de la dirección CGA, de forma que el valor del campo *Collision Count* se mantiene en cero.

Una vez obtenido el *Hash1* se procede a formar el identificador de la interfaz de red tomando en cuenta el valor del parámetro de seguridad (*Sec*) y realizando los cambios correspondientes en los bits “u” y “g” [10].

Finalmente, se forma la dirección CGA concatenando el prefijo de red con el identificador de la interfaz como se muestra en la Figura 4.6.

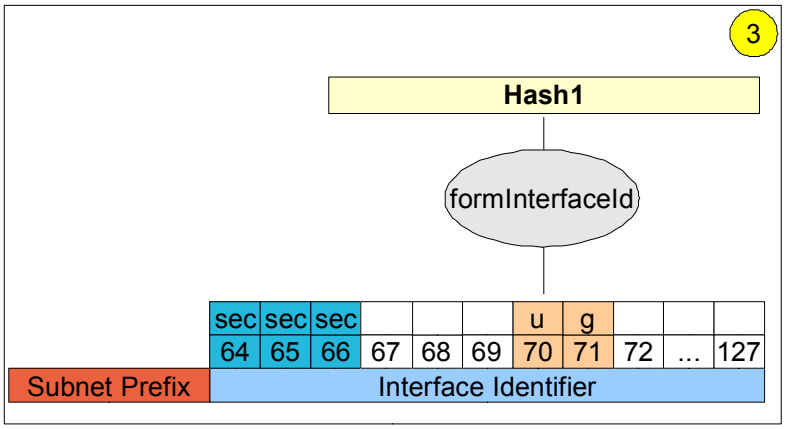

Figura 4.6: Dirección CGA

La clase ***RSASignature*** encapsula los métodos que permiten generar y verificar la firma digital contenida en la opción RSASignature como se define en [8], haciendo uso de la clase ***KeyManagement*** definida más adelante. En la Figura 4.7 se puede observar el diagrama de clase asociado.

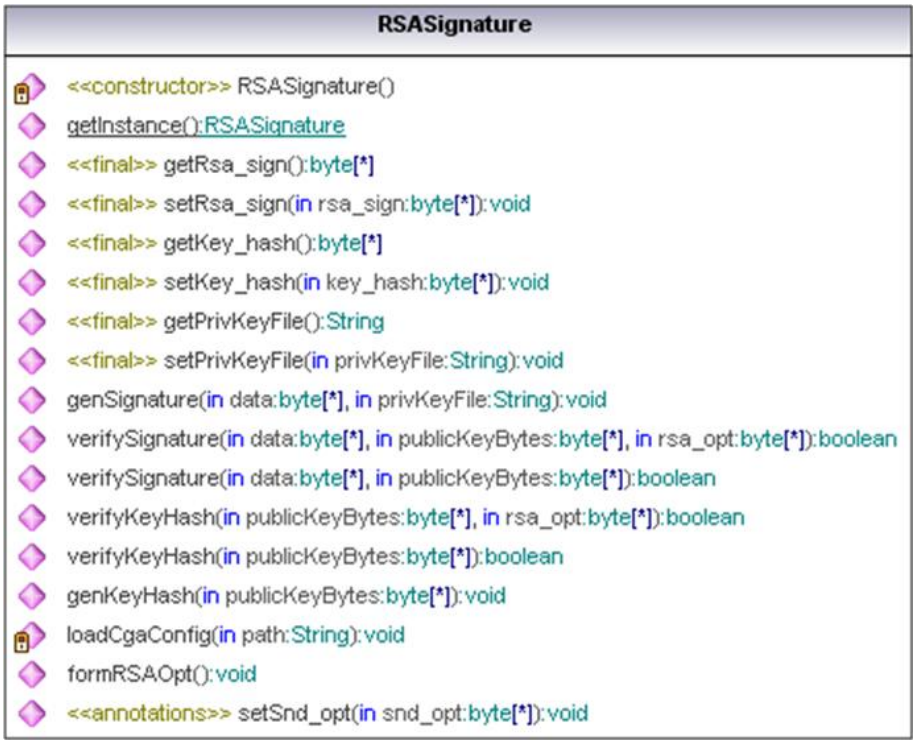

Figura 4.7: Diagrama de clase RSASignature

La generación de la firma digital requiere la contra parte de la clave pública, es decir, la clave privada; mientras que para la verificación de la firma sólo se requiere la clave pública del remitente y los datos firmados.

Es de hacer notar que la firma digital está directamente vinculada a la identificación del autor del mensaje, es decir, si la verificación falla existe la posibilidad de estar ante un ataque.

Por otra parte, la complejidad de la generación y la verificación de la firma dependen del tamaño de la clave utilizada y los datos de entrada. Es por tal razón, que esta opción es la última en ser generada en los mensajes salientes y verificada en los mensajes entrantes.

Además, si ocurre una falla durante la verificación de alguna otra opción SEND, no se requerirá ejecutar la verificación de la firma digital ya que el mensaje ha sido desechado y se pueden continuar procesando otros mensajes ND que siguen en la cola de ip6tables.

La clase ***Nonce*** contiene los métodos para generar la opción Nonce que será agregada en los mensajes de solicitud del protocolo ND. En la Figura 4.8 se puede observar el diagrama de clase.

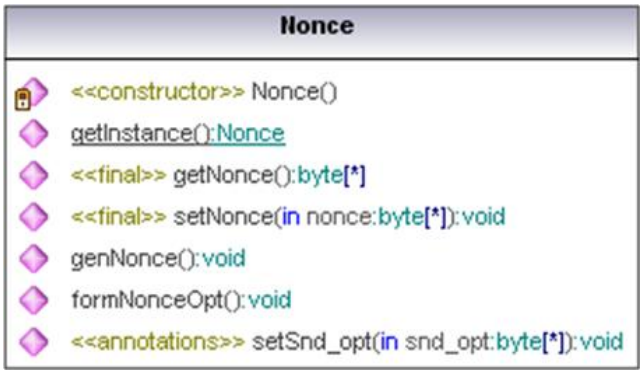

Figura 4.8: Diagrama de clase Nonce

La verificación del valor contenido en el campo *Nonce* (ver Figura 2.24) se realizará en la clase ***Monitor***, específicamente en el método *receivingRules,* tomando en cuenta el valor del campo *Nonce* recibido en respuesta a la solicitud realizada previamente y por el cual fue almacenado en la cache LRU (Least Recently Used). Si el mensaje recibido no fue previamente solicitado, no es necesario realizar la verificación.

La clase ***TimeStamp*** contiene los métodos que permiten generar y verificar la marca de tiempo de los mensajes ND. La Figura 4.9 muestra el diagrama de clase asociado.

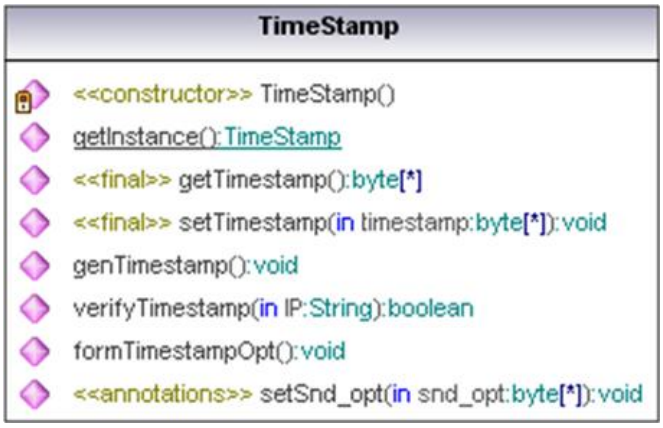

Figura 4.9: Diagrama de clase TimeStamp

Para generar el valor del campo *Timestamp* (ver Figura 2.23) se utiliza la clase ***Date*** que proporciona Java para obtener el tiempo UTC. Los primeros seis bytes más a la izquierda del campo representan los segundos; mientras que los dos últimos bytes más a la derecha ofrecen una precisión superior a los segundos, en este caso la precisión es del 1/64K=1/(64*1024) milisegundos.

Como se explicó en el Marco Teórico (ver Capítulo 2), para facilitar la verificación de los Timestamp, cada nodo debe almacenar el tiempo en que se recibió y aceptó el último mensaje ND asegurado por el protocolo SEND y por el cual es necesario incorporar una cache.

La clase ***TimestampCache*** encapsula los métodos que permiten verificar, agregar o eliminar las marcas de tiempos asociados a los nodos, que mantienen o mantuvieron una comunicación con el host. Esta clase ejecuta un hilo que se encarga de mantener la consistencia en la cache, verificando cada cierto tiempo (configurable) si alguna de las marcas de tiempo de los nodos ha expirado. En la Figura 4.10 se puede observar el diagrama de clase de la cache Timestamp.

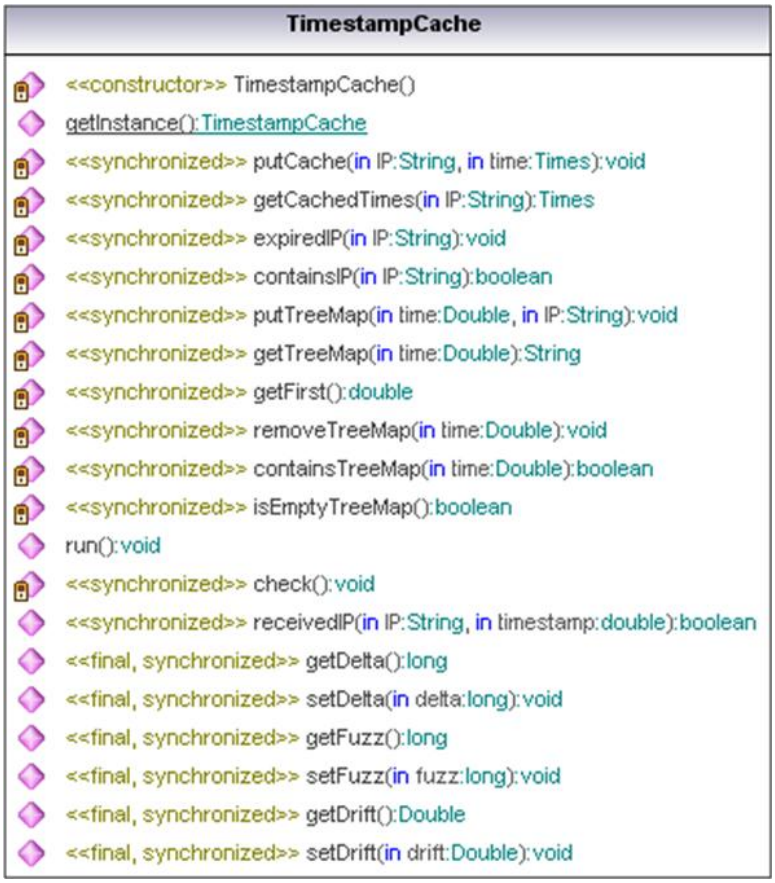

Figura 4.10: Diagrama de clase TimestampCache

La clase ***Monitor*** contiene los métodos que permiten la agregación o eliminación de las opciones SEND en los mensajes del protocolo ND. Se basa en las reglas definidas en las especificaciones SEND [8] para cada tipo de mensaje de solicitud o anuncio durante el proceso ND. Esta clase hace uso de la librería **"vserv-ipq-0.6.5.jar"** para capturar y manipular cada uno de los paquetes que llegan a la cola de ip6tables.

La Figura 4.11 muestra el diagrama de la clase Monitor.

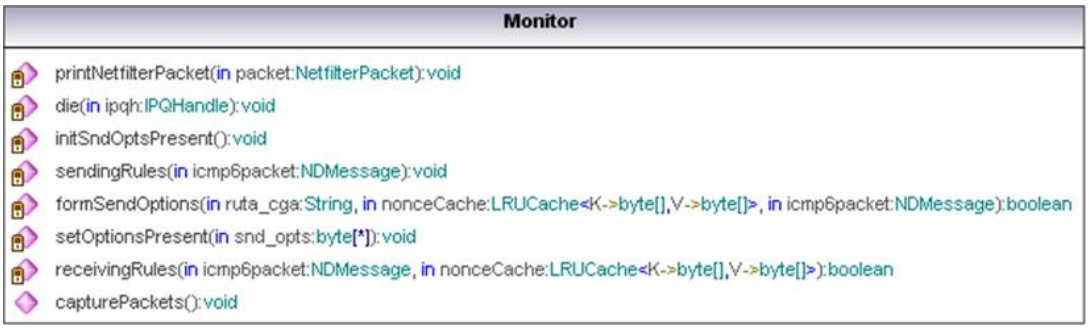

Figura 4.11: Diagrama de clase Monitor

La clase ***KeyManagent*** encapsula los métodos que permiten manejar todo lo relacionado con la recuperación, almacenamiento y generación de claves RSA, también incluye generación y verificación de firmas digitales RSASSAPKCS1-v1_5. La Figura 4.12 muestra el diagrama de clase asociado.

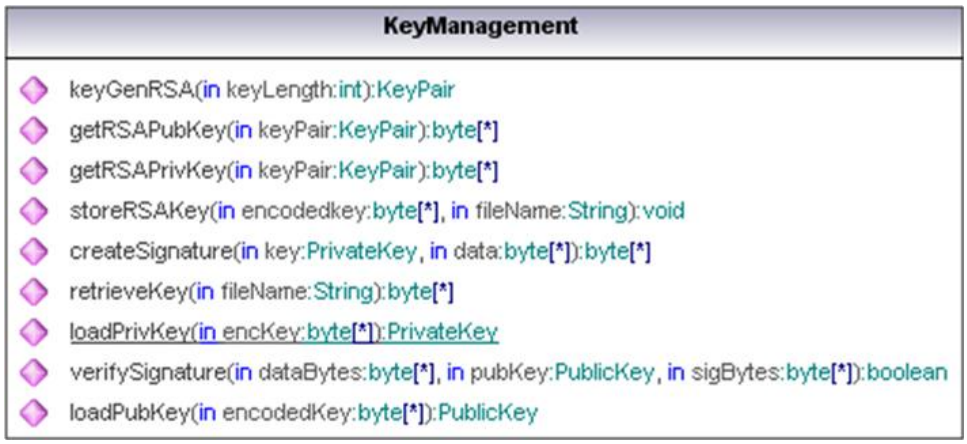

Figura 4.12: Diagrama de clase KeyManagement

Las clases ***Util*** y ***Files*** facilitan el manejo de archivos y conversión de datos como se observa en sus respectivos diagramas de clase mostrados en la Figura 4.13.

Figura 4.13: Diagrama de clases Files y Util

La clase ***NetInterfaces*** encapsula un método llamado verifyInterfaces que se encarga de verificar si la dirección IPv6 de tipo Link-Local del host es efectivamente una dirección CGA. En caso tal que la dirección sea inválida se generará y agregará una dirección CGA válida asociada a los parámetros CGA configurados en el host. En la Figura 4.14 se observa el diagrama de clase asociado.

Figura 4.14: Diagrama de clase NetInterfaces

La clase ***Configuration*** envuelve los métodos que permiten cargar la configuración inicial de la aplicación, la cual fue establecida previamente en un archivo de texto. Por otra parte, la clase ***Sendapp*** contiene el método principal de la aplicación y es donde se procede a llamar al resto de las clases que se requieren para ejecutar correctamente la aplicación. En la Figura 2.16 se muestra el diagrama de clase asociado a ambas clases.

Figura 4.15: Diagramas de clases Configuration y Sendapp

4.2 Fase de Desarrollo

El ambiente de desarrollo utilizado para escribir el código fuente fue *Eclipse Ganymade v3.4.0*[5], ya que provee excelentes herramientas de depuración y ayuda a mantener el código libre de errores sintácticos durante la escritura del mismo. Mientras que para la generación de los diagramas de clase se usó la herramienta *Altova UModel 2008*[6]

4.2.1 Requerimientos Mínimos del Sistema

Para el funcionamiento óptimo de la aplicación se tomaron en cuenta los siguientes requerimientos:

- Linux con núcleo 2.6 o posterior con soporte para IPv6.
- Java™ 2SE 1.6.0_07 o posterior.

[5] http://www.eclipse.org
[6] http://www.altova.com

- Wireshark 1.0.2 o cualquier otra herramienta de captura de tráfico que permita visualizar las opciones SEND en un mensaje ND.

4.2.2 Puesta en Marcha

Para la puesta en marcha, la aplicación no requiere instalación y es ejecutada a través de una shell, sólo es necesario el archivo **"sendapp.jar"** que se encuentra en el DVD anexo al documento. Si se desea generar o verificar direcciones CGA, sólo se necesita el archivo **"cgagen.jar"** que también se encuentra en dicho DVD.

Para poder ejecutar la aplicación se necesita tener permisos de ***root*** y tener previamente configurada las reglas de ip6tables tal como se muestra a continuación:

```
echo "Input"

ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "133"
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "134"
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "135"
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "136"
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "137"


echo "Output"

ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "133"
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "134"
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "135"
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "136"
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "137"
```

Con el comando **`ip6tables -L`** se puede verificar si las reglas fueron introducidas correctamente.

Con el comando **`ip6tables -F`** se pueden borrar todas las reglas de la tabla.

Cabe destacar que es necesario tener el archivo **"send.config"** bien configurado con las rutas a los archivos CGA y Logs.

Para ejecutar **"sendapp.jar"** se utiliza el siguiente comando:

```
java -Djava.library.path=<ruta_libvservipq.so> -jar sendapp.jar
-f <archivo_send.config>
```

Para ejecutar **"cgagen.jar"** se utiliza el siguiente comando:

```
java -jar cgagen.jar <opciones>
```

Ambas aplicaciones mantienen los pases de parámetros de la forma tradicional en Linux. Se puede usar la opción `--help` para obtener la ayuda correspondiente.

4.3 Fase de Pruebas

Durante la fase de pruebas se estudiaron diferentes casos. Primero fue necesario obtener un par de direcciones CGA que son utilizadas por los hosts del enlace. Utilizando la herramienta **"cgagen.jar",** se pueden obtener dichas direcciones como se muestra a continuación:

Dirección CGA para PC1

```
java -jar cgagen.jar -gen -f pc1.cga.params -k pc1.priv.key
-r 1024 -p 2004:: -s 1
2004::34cc:6bba:5947:d208
```

Dirección CGA para PC2

```
java -jar cgagen.jar -gen -f pc2.cga.params -k pc2.priv.key
-r 1024 -p 2004:: -s 1
2004::3869:5ddb:cdba:f44c
```

En la Figura 4.16 se muestra la topología de la red utilizada para realizar las pruebas, la cual está formada por dos hosts con sistema operativo Linux Ubuntu.

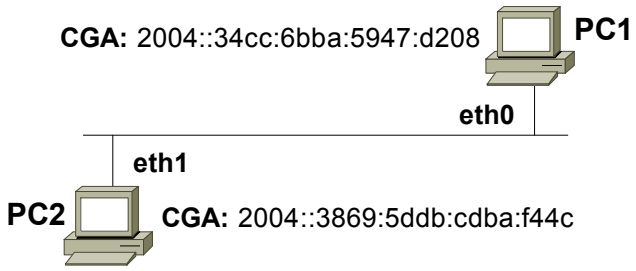

Figura 4.16: Topología de la red de pruebas

En la Tabla 4.1 se detallan las características de los hosts.

	SO	Int.	RAM	Java
PC1	Ubuntu 8.04 LTS	eth0	512 MB	1.6.0_07
PC2	Ubuntu 8.04 LTS	eth1	512 MB	1.6.0_07

Tabla 4.1: Características de los hosts de pruebas

Para un correcto funcionamiento de la aplicación, los hosts deben tener sus relojes sincronizados.

Caso de prueba 1: Comunicación entre dos hosts con **"sendapp.jar"**

Esta prueba consistió en establecer comunicación entre un par de hosts que ejecutan la aplicación SEND desarrollada a través de un ping entre el PC1 y PC2. La captura de pantalla que se observa en la Figura 4.17 muestra los mensajes ND intercambiados.

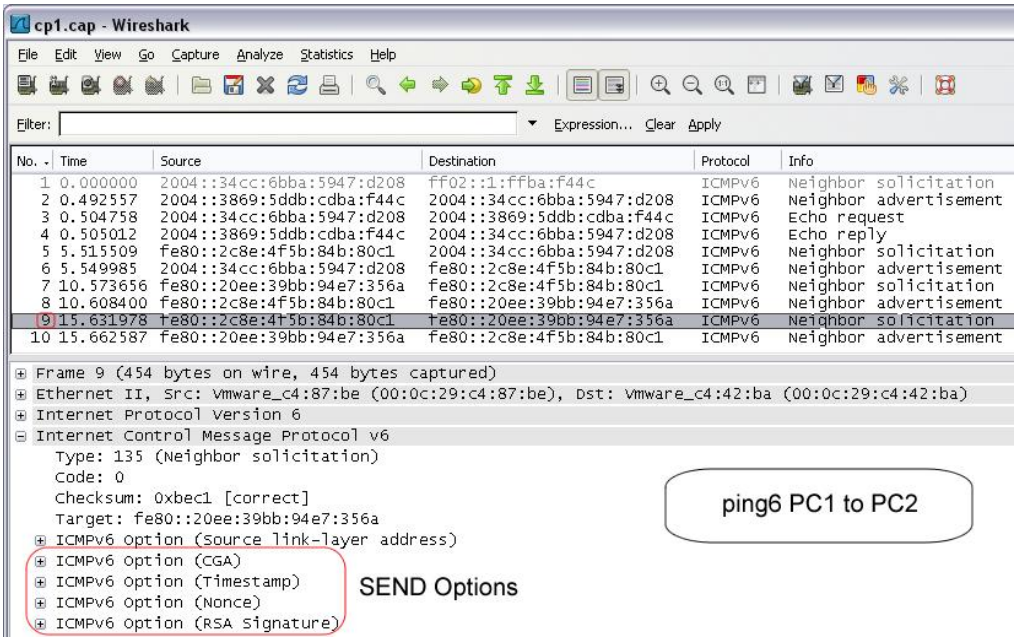

Figura 4.17: ping entre dos hosts con "sendapp" – (CP1)

A continuación se muestra parte de la traza dejada por el paquete entrante N° 9 de la captura mostrada en la Figura 4.17.

```
[06:53:07,577] INFO  Monitor  - <<----------------<<----------------<<<<
[06:53:07,577] INFO  Monitor  - NetfilterPacket Hook Local Input
[06:53:07,578] INFO  Monitor  - Incoming Interface: eth0
[06:53:07,579] INFO  Monitor  - Verificando opciones SEND...
[06:53:07,579] INFO  Monitor  - ICMPv6 Type: <- Neighbor Solicitation
[06:53:07,579] INFO  CGA  - Verificando CGA...
[06:53:07,580] INFO  CGA  - Verificando dirección:
fe80:0:0:0:2c8e:4f5b:84b:80c1

[06:53:07,580] DEBUG CGA  - CGA Parameters:

      1b be bd c4 5f 26 a4 c0 66 6b eb 46 73 44 fe 32
      fe 80 00 00 00 00 00 00 30 81 9f 30 0d 06 09
      2a 86 48 86 f7 0d 01 01 01 05 00 03 81 8d 00 30
      81 89 02 81 81 00 b0 89 3b 51 88 d3 f2 3c 9b a0
      98 4f 99 01 b4 5f de 81 89 b3 d6 ba 96 ff a7 05
      b3 49 4c 28 ce 60 2e ba e7 f6 76 f4 c2 87 f9 51
      36 be 08 12 ab c2 e5 00 50 8a b5 be fb 04 79 99
      79 b4 ee 88 5b 45 b7 f1 88 a9 e2 e4 52 3a 75 9b
      ea 94 87 38 56 49 8d 63 66 b6 78 1e d0 41 4a f5
      25 e9 cf 69 50 9b f2 b9 68 91 a9 8c e6 8a 6f 61
      8f 45 8d 40 6a fc a4 26 ae 1e a8 7d 8b 48 88 c2
```

```
      20 b3 c3 e3 93 2b 02 03 01 00 01
[06:53:07,581] INFO  CGA  - Verificación CGA exitosa!
[06:53:07,581] INFO  TimeStamp  - Verificando Timestamp...
[06:53:07,581] INFO  TimestampCache  - 1.223292216E9 > 1.223292209E9 ?
[06:53:07,581] INFO  RSASignature  - Verificando KeyHash...
[06:53:07,582] INFO  RSASignature  - Keyhash verificado exitosamente...
[06:53:07,582] INFO  RSASignature  - Verificando firma digital...
[06:53:07,585] INFO  RSASignature  - Firma verificada exitosamente...
[06:53:07,585] DEBUG Monitor  - Paquete Modificado

      60 00 00 00 00 20 3a ff fe 80 00 00 00 00 00 00
      2c 8e 4f 5b 08 4b 80 c1 fe 80 00 00 00 00 00 00
      20 ee 39 bb 94 e7 35 6a 87 00 7b a6 00 00 00 00
      fe 80 00 00 00 00 00 00 20 ee 39 bb 94 e7 35 6a
      01 01 00 0c 29 c4 87 be

[06:53:07,585] INFO  Monitor  - Reinjected!!!
```

Una vez que la firma digital es verificada exitosamente, se procede a eliminar las opciones SEND del mensaje ND y el paquete es reinyectado y verificado por el núcleo del sistema sólo con las opciones del mensaje ND correspondiente, es decir, que el procesamiento previo del mensaje asegurado es completamente transparente para el núcleo.

Caso de prueba 2: Comunicación entre un host SEND y no SEND

En este caso de prueba, PC1 es un host que implementa SEND; mientras que PC2 no. Ambas direcciones IPv6 configuradas en cada host se mantienen, ya que es completamente transparente para PC2 que su dirección sea CGA o no.
El ping se realiza desde PC2 y cabe destacar que no hay reglas configuradas en la ip6tables, ya que no se encuentra ninguna aplicación procesando los paquetes capturados. A continuación se muestra parte de la traza dejada por PC1 en su log.

```
[18:20:09,834] INFO  Monitor  - <<----------------<<----------------<<<<
[18:20:09,834] INFO  Monitor  - NetfilterPacket Hook Local Input
[18:20:09,835] INFO  Monitor  - Incoming Interface: eth0
[18:20:09,835] INFO  Monitor  - Verificando opciones SEND...
[18:20:09,835] INFO  Monitor  - ICMPv6 Type: <- Neighbor Solicitation
[18:20:09,835] ERROR Monitor  - No se encontraron opciones SEND en el
paquete. Paquete desechado!!!
[18:20:09,835] INFO  Monitor  - Dropped!!!
```

Finalmente, como se observa en la traza, el paquete entrante es **desechado** ya que no se encontraron opciones SEND dentro del mensaje ND. Esta versión de la aplicación sólo permite la comunicación entre nodos asegurados con SEND.

Caso de prueba 3: DAD

En este caso de prueba, PC1 y PC2 fueron configurados con las mismas direcciones CGA. Esta versión de la aplicación no implementa el algoritmo DAD por la complejidad que éste acarrea. La duplicidad es detectada a nivel del núcleo del sistema operativo.

La dirección CGA utilizada será: 2004::34cc:6bba:5947:d208

La Figura 2.19 muestra la captura DAD ocurrida entre PC1 y PC2

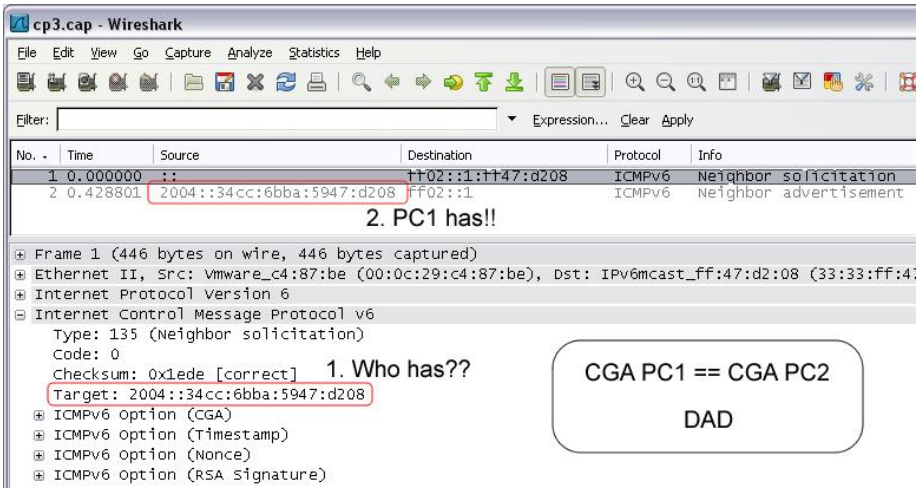

Figura 4.18: DAD – (CP3)

Caso de prueba 4: Verificación de la Firma Digital

En este caso de prueba, PC2 fue configurado con una clave privada que no corresponde con la clave pública utilizada para generar la dirección CGA. A continuación se muestra parte de la traza dejada por PC1 en su log.

```
[07:07:29,368] INFO  Monitor  - <<----------------<<----------------<<<<
[07:07:29,368] INFO  Monitor  - NetfilterPacket Hook Local Input
[07:07:29,369] INFO  Monitor  - Incoming Interface: eth0
[07:07:29,369] INFO  Monitor  - Verificando opciones SEND...
[07:07:29,369] INFO  Monitor  - ICMPv6 Type: <- Neighbor Advertisement
[07:07:29,369] INFO  CGA  - Verificando CGA...
[07:07:29,369] INFO  CGA  - Verificando dirección:
2004:0:0:0:3869:5ddb:cdba:f44c
[07:07:29,371] INFO  CGA  - Verificación CGA exitosa!
[07:07:29,371] INFO  TimeStamp  - Verificando Timestamp...
[07:07:29,372] DEBUG TimestampCache  - Host nuevo
2004:0:0:0:3869:5ddb:cdba:f44c
[07:07:29,372] INFO  TimestampCache  - Time now: 1223293044 Time recv:
1.223293077E9 Diff: -33.0 Delta: 300
[07:07:29,372] DEBUG TimestampCache  - Insertando en la cache:
2004:0:0:0:3869:5ddb:cdba:f44c time: 1.223293077E9
[07:07:29,373] DEBUG Monitor  - Nonce recibido:7d 4f 3b c6 5a ea
[07:07:29,378] INFO  RSASignature  - Verificando KeyHash...
[07:07:29,379] INFO  RSASignature  - Keyhash verificado exitosamente...
[07:07:29,380] INFO  RSASignature  - Verificando firma digital...
[07:07:29,383] ERROR Monitor  - Verificación RSA Signature no exitosa.
Desechando el paquete... Paquete desechado!!!

[07:07:29,384] INFO  Monitor  - Dropped!!!
```

PC2 ignora por completo que está utilizando una clave privada para firmar los mensajes salientes que no corresponden con su clave pública. Por otra parte, verificar la correspondencia entre las claves puede ser costoso en tiempo de ejecución.

Caso de prueba 5: Dirección Scope Global no CGA

En este caso de prueba, PC1 fue configurado con una dirección no CGA, por lo que no pudo comunicarse con PC2 ya que el paquete fue desechado luego de la respectiva verificación de la dirección. A continuación se muestra parte de la traza de PC1 dejada en su log.

```
[20:11:54,683] INFO  Monitor  - >>>>---------------->>---------------->>
[20:11:54,684] INFO  Monitor  - NetfilterPacket Hook Local Output
[20:11:54,685] INFO  Monitor  - Outgoing Interface: eth0
[20:11:54,686] INFO  Monitor  - Agregando opciones SEND...
[20:11:54,687] INFO  CGA  - Verificando CGA...
[20:11:54,687] INFO  CGA  - Verificando dirección: 2004:0:0:0:0:0:0:1
[20:11:54,689] DEBUG CGA  - CGA Parameters:

      35 91 9b aa e5 04 97 5b 9b e4 78 1b c8 41 6f 18
      20 04 00 00 00 00 00 00 30 81 9f 30 0d 06 09
      2a 86 48 86 f7 0d 01 01 01 05 00 03 81 8d 00 30
      81 89 02 81 81 00 c9 06 49 c1 7b 65 02 2e b0 08
      d4 fa e4 82 d3 52 ee 8b 79 c0 d0 64 93 61 a2 84
      3d 25 0b 32 9b 2b de f0 a8 e8 11 55 ae 96 fb a2
      35 0d c4 df ea ba df 83 f5 a0 09 d0 00 3a 0c 19
      b6 94 13 d6 64 fe dc 14 bf 08 a3 b5 45 f3 90 e9
      87 3d 22 20 e5 cf ce ac e6 8d f2 75 8c ff d0 6b
      a0 46 13 eb 5e ce 3f 16 aa 82 dc 5b 42 52 59 ea
      c9 62 53 5c 1e c5 ce d0 30 ca 58 08 1f 63 99 f6
      b4 78 8b 2e 5e fb 02 03 01 00 01

[20:11:54,691] ERROR CGA  - Verificando hash1 e ignorando bits (sec y
u/g)...Falló
[20:11:54,691] ERROR Monitor  - Verificación CGA no exitosa. Desechando el
paquete... Paquete desechado!!!

[20:11:54,691] INFO  Monitor  - Dropped!!!
```

Caso de prueba 6: Comunicación entre un host **"sendapp.jar"** y un host **"sendd"**

En este caso de prueba, PC1 fue configurado para usar "sendapp.jar"; mientras que PC2 se configuró con la aplicación SEND desarrollada por DoCoMo's.

En tal sentido, PC2 fue configurada con una dirección CGA generada a partir del archivo de parámetros CGA de muestra que contiene la aplicación SEND de DoCoMo's.

A continuación, para obtener una dirección CGA con "cgatool" a partir del archivo de parámetros CGA se utilizó el siguiente comando:

```
cgatool --gen -D mn1.cga.params -k mn1.key.pem -o /tmp/der -s 1
-p 2004::
2004::343d:3030:2537:372d
```

Luego, para ejecutar “sendd” en modo debug se utilizó el siguiente comando:

```
sendd -f -c <ruta_sendd.conf> -d -d -d
```

La Figura 4.19 muestra la captura de los mensajes ND entre PC1 y PC2

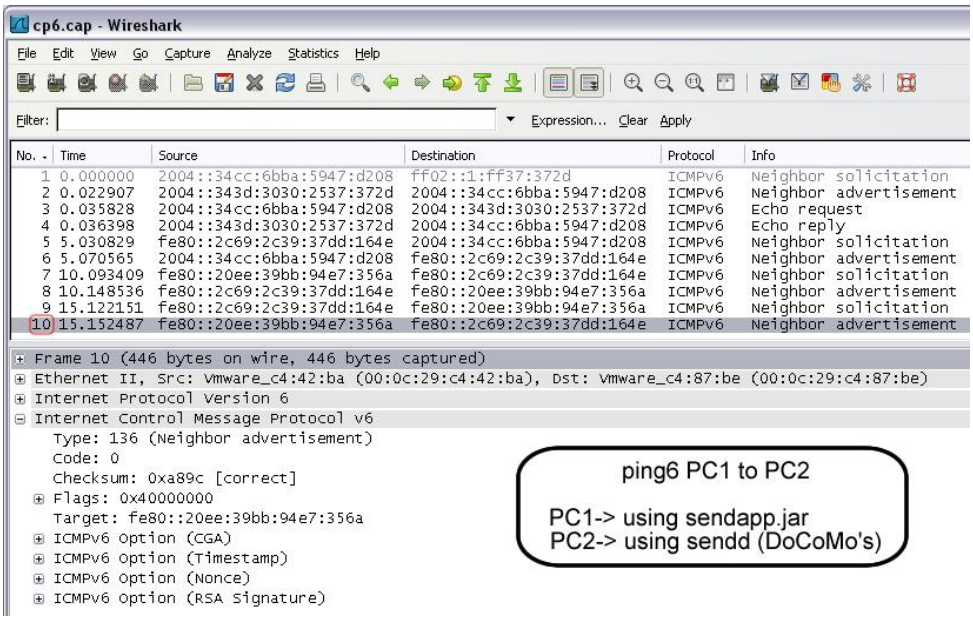

Figura 4.19: ping entre un host “sendapp” y un host “sendd” – (CP6)

A continuación se muestra parte de la traza obtenida de la aplicación SEND de DoCoMo’s al paquete N° 10 de la captura.

```
[Oct 01 14:20:35] sendd: handle_incoming: Incoming packet uses signature
method 'rfc3971'
[Oct 01 14:20:35] sendd: verify_cga: CGA verification for
fe80::20ee:39bb:94e7:356a
[Oct 01 14:20:35] sendd: cga_verify: DER-encoded data:
      35 91 9b aa e5 04 97 5b  9b e4 78 1b c8 41 6f 18
      fe 80 00 00 00 00 00 00  00 30 81 9f 30 0d 06 09
      2a 86 48 86 f7 0d 01 01  01 05 00 03 81 8d 00 30
      81 89 02 81 81 00 c9 06  49 c1 7b 65 02 2e b0 08
      d4 fa e4 82 d3 52 ee 8b  79 c0 d0 64 93 61 a2 84
      3d 25 0b 32 9b 2b de f0  a8 e8 11 55 ae 96 fb a2
      35 0d c4 df ea ba df 83  f5 a0 09 d0 00 3a 0c 19
      b6 94 13 d6 64 fe dc 14  bf 08 a3 b5 45 f3 90 e9
      87 3d 22 20 e5 cf ce ac  e6 8d f2 75 8c ff d0 6b
      a0 46 13 eb 5e ce 3f 16  aa 82 dc 5b 42 52 59 ea
      c9 62 53 5c 1e c5 ce d0  30 ca 58 08 1f 63 99 f6
      b4 78 8b 2e 5e fb 02 03  01 00 01
```

```
[Oct 01 14:20:35] sendd: libcga: cga_verify: Address verified
[Oct 01 14:20:35] sendd: verify_cga: ok
[Oct 01 14:20:35] sendd: snd_check_timestamp: 1222887013.00004 >
1222887011.00960 ?
[Oct 01 14:20:35] sendd: snd_check_timestamp: timestamp: ok
[Oct 01 14:20:35] sendd: incoming_thr: verify sig: ok
[Oct 01 14:20:35] sendd: snd_timestamp_cache_upd:
[Oct 01 14:20:35] sendd: incoming_thr: delivering pkt (272 bytes)
[Oct 01 14:20:35] sendd: libtimer: timer_check: now 1222887035.538877 pqmax
1222887037.419235
```

En este capítulo se pudo observar el correcto funcionamiento de la aplicación SEND desarrollada a través de los diferentes casos de prueba, brindando la seguridad que los hosts requieren durante el proceso de descubrimeinto de vecinos. Además, la efectiva comunicación entre hosts que ejecutan diferentes implementaciones de SEND (ver Caso de prueba 6) evidencia la correcta definición del protocolo.

Por último, es importante señalar que todos los casos de pruebas con sus respectivas capturas y logs podrán ser encontrados en el DVD anexo al documento. También se cuenta con un *Virtual Appliance* el cual tiene instalado y configurado la aplicación SEND de DoCoMo's y una carpeta llamada SeND_UCV que contiene la aplicación **"sendapp.jar"** y **"cgagen.jar"** listos para ejecutar y probar.

5. Conclusiones y Trabajos Futuros

Los aspectos de seguridad en una red IPv6 no son en esencia diferentes a los de una red IPv4, por lo tanto, muchos de los ataques de IPv4 se pueden realizar con IPv6 y en el caso de ND (Neighbor Discovery) esta posibilidad está abierta.

En este sentido, SEND permite minimizar y contrarrestar los ataques y amenazas en los que se ve envuelto un nodo IPv6 durante el proceso ND; pero a diferencia de IPsec resulta más práctico y sencillo de implementar.

Es de hacer notar que SEND hace uso de la criptografía asimétrica en las capas inferiores, lo cual es una idea innovadora que cambia el concepto que se tiene de la protección de los datos. Así pues, una dirección IPv6 es creada en función de una clave pública, que luego puede ser utilizada para corroborar la autenticidad de un nodo a través de la verificación de la firma digital emitida por su contraparte (clave privada).

También, con SEND es posible tener una autoridad de confianza, la cual permitirá determinar la función de los nodos en el enlace y con ello, ofrecer mayor seguridad al entorno.

En lo que respecta al desarrollo de la aplicación, se logró incluir dentro de los mensajes ND las opciones SEND requeridas para brindar la protección deseada durante el proceso de descubrimiento de vecinos. Adicionalmente, se puede generar y verificar direcciones CGA a través de la herramienta desarrollada.

La aplicación SEND desarrollada podrá ser utilizada como caso de estudio y permite sentar las bases para trabajos futuros y ser extendida con nuevas funcionalidades, gracias al diseño con el cual fue concebida, dando la oportunidad a que otros investigadores puedan profundizar, mejorar o desarrollar otros módulos. Algunos trabajos complementarios que enriquecerían de gran manera la aplicación son:

- Incluir el proceso de Router Discovery a través del mecanismo de confianza ADD [8].
- Agregar extensiones al formato CGA que puedan dar soporte a múltiples resúmenes hash [12].
- Desarrollar una versión para sistemas operativos con ambiente Microsoft Windows.

Por otra parte, la implementación de SEND en el espacio de usuario del sistema operativo Linux no debe ser considerado como una solución final; pues un desarrollo al nivel del núcleo permitiría optimizar la agregación y verificación de las opciones dentro de los mensajes ND y el manejo de las caches.

En definitivo, esta implementación de SEND brinda la posibilidad de ampliar los conocimientos que se tienen dentro del escaso mundo de aplicaciones IPv6, en particular del protocolo SEND y su complemento CGA.

Apéndice A: Manual de Usuario

Introducción

Esta es una aplicación open source que permite brindar seguridad a un enlace IPv6 a través del protocolo SEND (Secure Neighbor Discovery), definido en la RFC 3971. También incluye una herramienta que permite la generación y verificación de la dirección CGA (Cryptographically Generated Addresses), definida en la RFC 3972.

Requisitos Previos

- Linux con núcleo 2.6 o posterior con soporte para IPv6.
- Java™ 2SE 1.6.0_07 o posterior.
- Wireshark 1.0.2 o cualquier otra herramienta de captura de tráfico que permita visualizar las opciones SEND en un mensaje ND.

Puesta en Marcha de la Aplicación

La aplicación no requiere instalación y se ejecuta a través de una shell. Sólo se necesita el archivo ejecutable "sendapp.jar" que se encuentra contenido en el archivo comprimido "SeND_UCV.tar.gz".

El archivo "SeND_UCV.tar.gz" debe contener:

- Un directorio principal llamado "SeND_UCV" y dos directorios internos llamados "cgagen" y "sendapp".
- El directorio "cgagen" contiene la herramienta para la generación y verificación de direcciones CGA llamada "cgagen.jar", un archivo de ejemplo y un subdirectorio con el código fuente llamado "src".
- El directorio "sendapp" contiene el archivo "sendapp.jar" y los subdirectorios "cga", "lib", "logs", "src" y el archivo de configuración "send.config".

Antes de ejecutar la aplicación es necesario tener permisos de ***root***, y tener previamente configuradas las reglas de ip6tables tal como se muestra a continuación:

Encolando todos los paquetes entrantes:

```
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "133"
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "134"
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "135"
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "136"
ip6tables -A INPUT -p icmpv6 -j QUEUE --icmpv6-type "137"
```

Encolando todos los paquetes salientes:

```
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "133"
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "134"
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "135"
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "136"
ip6tables -A OUTPUT -p icmpv6 -j QUEUE --icmpv6-type "137"
```

Con el comando **`ip6tables -L`** se puede verificar si las reglas fueron introducidas correctamente.

Con el comando **`ip6tables -F`** se pueden borrar todas las reglas de la tabla.

Configuración

En esta versión de la aplicación, los hosts no participan en el descubrimiento de los routers que residen en el enlace, por lo que es necesario cumplir con los siguientes pasos:

1. Generación de la dirección CGA y su archivo de parámetros asociado.
2. Configuración del nivel de traza de la aplicación (Log).
3. Configuración sendapp.jar

1. Generación de la dirección CGA y su archivo de parámetros asociado

Se debe generar el archivo de parámetros CGA y al menos una dirección IPv6 del tipo CGA, para ello se utiliza la herramienta "cgagen.jar" a través del siguiente comando.

```
java -jar cgagen.jar -gen -f pc1.cga.params -k pc1.priv.key
-r 1024 -p 2004:: -s 1
2004:: 34cc:6bba:5947:d208
```

Como resultado de la ejecución del comando anterior se obtiene: un archivo de parámetros CGA llamado **"pc1.cga.params"**, un archivo contenedor de la clave privada RSA llamado **"pc1.priv.key"** cuyo tamaño de clave es de 1024 bits y por último la dirección CGA con prefijo de red 2004 utilizando un parámetro de seguridad igual a uno (*Sec = 1*).

Para generar una nueva dirección CGA a partir del archivo de parámetros anteriormente generado se utiliza el siguiente comando:

```
java -jar cgagen.jar -gen -f pc1.cga.params -p <prefix>
-s <sec>
```

2. Configuración del nivel de traza de la aplicación (Logs)

El directorio “logs” contiene un archivo de configuración llamado “log4j.config”, el cual puede ser editado para manejar el nivel de traza que se requiere de la aplicación. Por defecto, sólo se muestran mensajes informativos y mensajes de error; también serán almacenados dichos mensajes en un archivo de configuración llamado “send.log” en el directorio home (/home).

3. Configuración “sendapp.jar”

La configuración de “sendapp.jar” requiere leer un archivo comúnmente llamado “send.config”. Dicho archivo deberá contener las rutas a los archivos de configuración CGA y Logs, así como los valores por defecto que eventualmente pondrán ser modificados.

El archivo mantiene un formato de *clave=valor*. Las líneas que comienzan con “#” son comentarios.

En el directorio “sendapp” se encuentra un archivo llamado “send.config” de muestra. Las claves siguientes son obligatorias:

```
CGA_FILES=<ruta_archivo_configuración_cga>
LOG_FILES=<ruta_archivo_log4j.config>
```

El archivo de configuración CGA generalmente llamado “cga.config” sigue la misma estructura *clave=valor* y contiene las rutas al archivo de parámetros CGA, y el archivo que contiene la clave privada del host. También contiene el valor del parámetro de seguridad usado por defecto.

Las claves siguientes son obligatorias en el archivo “cga.config”:

```
cga_params=<ruta_archivo_cga.params>
cga_priv_key =<ruta_archivo_priv.key>
cga_sec=<valor_entero>
```

Ejecutando “sendapp.jar”

Para ejecutar la aplicación, sólo es necesario introducir el siguiente comando en modo ***root***:

```
java -Djava.library.path=<ruta_libvservipq.so> -jar sendapp.jar
-f <archivo_send.config>
```

La librería dinámica **"libvservipq.so"** se encuentra en el directorio llamado "lib" y es esencial para el correcto funcionamiento de la aplicación.

Limitaciones

Esta aplicación no implementa el proceso de Router Discovery y en consecuencia, solamente los mensajes de Neighbor Advertisement y Neighbor Solicitation del protocolo Neighbor Discovery serán procesados.

Todos aquellos mensajes no asegurados con las opciones SEND serán desechados.

Además, esta implementación de SEND es independiente del núcleo del sistema operativo, por lo que no se tiene control sobre la cache de vecinos, es decir, que los mensajes NS o NA que estén fuera del valor TIMESTAMP_DELTA permitido serán desechados aunque sigan vigentes en la cache de vecino; así pues, es necesario que el reloj en los sistemas esté totalmente sincronizado o que la diferencia de tiempo no sobrepase el valor DELTA definido en el archivo de configuración "**send.config**". Una sincronización de forma automática a través de un servidor NTP también puede ser considerada.

Apéndice B: Virtual Appliance

Las *Virtual Appliances* son máquinas virtuales preconfiguradas y listas para funcionar que pueden ser utilizadas en un entorno plenamente productivo o de prueba.

La distribución tradicional del software suele requerir que los clientes instalen y configuren el software en su hardware de manera personalizada. Incluso cuando se trata de un usuario con conocimientos técnicos elevados, este proceso suele implicar una inversión en tiempo alta. Es por ello que la aplicación desarrollada se distribuye embebida en una máquina virtual Ubuntu 8.04 LTS[7], con un mínimo de paquetes instalados permitiendo así su portabilidad y distribución.

Adicionalmente, dentro de la *Virtual Appliance* se encuentra instalada y configurada la aplicación SEND desarrollada por DoCoMo's y el analizador de tráfico Wireshark[8].

Para poder utilizar la máquina virtual es necesario tener instalado un *VMware Player* que puede ser descargado del sitio oficial de VMware[9] o tener *VMware Workstation* 6.0.4 build 93057 o superior.

Usuario: pc
Contraseña: abc

Por cuestiones de seguridad la cuenta de usuario ***root*** está desactivada en Ubuntu por defecto. En su lugar, cada vez que se requiere ejecutar un comando con permisos de ***root***, se tiene que anteponer el comando *sudo*. Por ejemplo: **sudo** <comando>

Sin embargo, cuando se necesita realizar muchas tareas administrativas, se torna tedioso escribir decenas de veces "sudo"; y es por ello, que la cuenta ***root*** ha sido activada en esta máquina virtual.

Para ejecutar comandos como ***root***, sólo es necesario escribir el comando "su" e introducir la contraseña "abc" **.** Por ejemplo:

```
pc@pc:~$ su
Contraseña:
root@pc:/home/pc#
```

Requerimientos mínimos del sistema:

- 512 MB de RAM disponible (si posee VMware Workstation puede reducir el tamaño de la memoria).
- 4 GB de espacio en disco duro.

[7] http://www.ubuntu.com
[8] http://www.wireshark.org
[9] http::/www.vmware.com

Apéndice C: Generando una Dirección CGA

El objetivo principal de este apéndice es detallar el proceso de generación de una dirección CGA.

Datos de entrada:
Parámetro de Seguridad (*Sec*)= 1
Prefijo de red (*Subnet Prefix*)= 2004::
Clave pública (*Public key*):

```
3081 9f30 0d06 092a 8648 86f7 0d01 0101 0500 0381 8d00 3081
8902 8181 008d c682 ba21 cdfa 06ae 1a4a 831f af80 c8c6 a761
eabe f357 d939 9ca5 6e22 df01 432e 5b01 6f5d 886c 4b7d efa3
c98b 1c2e 4a39 3deb eff8 6ac2 354e 562f ae5c ee4f 165e 0653
6e6a e811 ec88 fe72 f72f dc57 4ef6 6711 e01d 6462 fca2 e7bd
7939 4acf b98c 4dd0 d4bc a343 49f8 8723 39fd ae0e bf5e c0fe
cc9c ae2d 2e48 db1d 92ee 9cb7 8b02 0301 0001
```

El modificador (*Modifier*) es inicializado con un valor generado de forma aleatoria:

Modifier:

```
18fa f0a4 997c 0ace c672 09d4 9286 8b50
```

La entrada de parámetros para el cálculo del *Hash2* es:

```
18fa f0a4 997c 0ace c672 09d4 9286 8b50
0000 0000 0000 0000 00
3081 9f30 0d06 092a 8648 86f7 0d01 0101 0500 0381 8d00 3081
8902 8181 008d c682 ba21 cdfa 06ae 1a4a 831f af80 c8c6 a761
eabe f357 d939 9ca5 6e22 df01 432e 5b01 6f5d 886c 4b7d efa3
c98b 1c2e 4a39 3deb eff8 6ac2 354e 562f ae5c ee4f 165e 0653
6e6a e811 ec88 fe72 f72f dc57 4ef6 6711 e01d 6462 fca2 e7bd
7939 4acf b98c 4dd0 d4bc a343 49f8 8723 39fd ae0e bf5e c0fe
cc9c ae2d 2e48 db1d 92ee 9cb7 8b02 0301 0001
```

Los 112 bits del resumen *Hash2* calculado son `ea5e c7a7 5b35 f192 0016 71e6 af9e` pero los primeros 16 bits no comienzan en cero (16**Sec* = 16*1 = 16), es decir, hay que incrementar el valor del modificador en uno.

Luego de 24614 iteraciones, se obtiene el *Hash2* cuyo valor es `0000 53a5 0e3a d738 05e8 2a2c 2735` y el modificador final es `18fa f0a4 997c 0ace c672 09d4 9286 eb75`. Mientras mayor sea el valor del parámetro *Sec* usado, el tiempo de procesamiento para encontrar el resumen que cumpla con la condición deseada aumentará.

Para obtener el *Hash1* se tiene como parámetros de entrada:

```
18fa f0a4 997c 0ace c672 09d4 9286 eb75
2004 0000 0000 0000
00
3081 9f30 0d06 092a 8648 86f7 0d01 0101 0500 0381 8d00 3081
8902 8181 008d c682 ba21 cdfa 06ae 1a4a 831f af80 c8c6 a761
```

```
eabe f357 d939 9ca5 6e22 df01 432e 5b01 6f5d 886c 4b7d efa3
c98b 1c2e 4a39 3deb eff8 6ac2 354e 562f ae5c ee4f 165e 0653
6e6a e811 ec88 fe72 f72f dc57 4ef6 6711 e01d 6462 fca2 e7bd
7939 4acf b98c 4dd0 d4bc a343 49f8 8723 39fd ae0e bf5e c0fe
cc9c ae2d 2e48 db1d 92ee 9cb7 8b02 0301 0001
```

Los 64 bits del resumen (*Hash1*) calculado son `21ed 8aa5 8506 ca26` y son usados para formar el identificador de la interfaz (Interface ID).

Al introducir el valor *Sec* en los 3 primeros bits del *Hash1* y colocando en cero los bits “u” y “g”, se tiene el identificador de la interfaz cuyo valor es `20ed 8aa5 8506 ca26`.

Finalmente, se forma la dirección CGA uniendo el prefijo de red con el identificador de la interfaz `2004::20ed:8aa5:8506:ca26` y se crea el archivo de parámetros CGA con los siguientes datos:

```
18 fa f0 a4 99 7c 0a ce c6 72 09 d4 92 86 eb 75
20 04 00 00 00 00 00 00 30 81 9f 30 0d 06 09
2a 86 48 86 f7 0d 01 01 01 05 00 03 81 8d 00 30
81 89 02 81 81 00 8d c6 82 ba 21 cd fa 06 ae 1a
4a 83 1f af 80 c8 c6 a7 61 ea be f3 57 d9 39 9c
a5 6e 22 df 01 43 2e 5b 01 6f 5d 88 6c 4b 7d ef
a3 c9 8b 1c 2e 4a 39 3d eb ef f8 6a c2 35 4e 56
2f ae 5c ee 4f 16 5e 06 53 6e 6a e8 11 ec 88 fe
72 f7 2f dc 57 4e f6 67 11 e0 1d 64 62 fc a2 e7
bd 79 39 4a cf b9 8c 4d d0 d4 bc a3 43 49 f8 87
23 39 fd ae 0e bf 5e c0 fe cc 9c ae 2d 2e 48 db
1d 92 ee 9c b7 8b 02 03 01 00 01
```

Apéndice D: Analizando un Paquete con Opciones SEND

El objetivo principal de este apéndice es mostrar con mayor detenimiento la interpretación de los bytes en la captura de un mensaje NS y NA que contienen opciones SEND.

A continuación se analizan los datos correspondientes a los paquetes 1 y 2 de la captura realizada en el caso de prueba 1 (CP1) descrito en el Marco Metodológico (ver Capítulo 3). También se podrán encontrar dichas capturas en el directorio "casos de prueba" en el DVD anexo al documento.

A. Analizando un mensaje NS saliente

Antes de analizar la captura correspondiente al mensaje NS, hay que tener en cuenta que el host tiene preconfigurada una dirección CGA, así como sus respectivos archivos asociados: archivo de parámetros CGA y archivo contenedor de la clave privada.

A continuación se muestra el flujo de bytes asociado al paquete ICMPv6 N°1 del caso de prueba 1.

Dirección fuente: `2004::34cc:6bba:5947:d208`
Dirección destino: `ff02::1:ffba:f44c`

```
0000   33 33 ff ba f4 4c 00 0c 29 c4 42 ba 86 dd 60 00
0010   00 00 01 90 3a ff 20 04 00 00 00 00 00 00 34 cc
0020   6b ba 59 47 d2 08 ff 02 00 00 00 00 00 00 00 00
0030   00 01 ff ba f4 4c 87 00 dd c4 00 00 00 00 20 04
0040   00 00 00 00 00 00 38 69 5d db cd ba f4 4c 01 01
0050   00 0c 29 c4 42 ba 0b 18 01 00 35 91 9b aa e5 04
0060   97 5b 9b e4 78 1b c8 41 6f 18 20 04 00 00 00 00
0070   00 00 00 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d
0080   01 01 01 05 00 03 81 8d 00 30 81 89 02 81 81 00
0090   c9 06 49 c1 7b 65 02 2e b0 08 d4 fa e4 82 d3 52
00a0   ee 8b 79 c0 d0 64 93 61 a2 84 3d 25 0b 32 9b 2b
00b0   de f0 a8 e8 11 55 ae 96 fb a2 35 0d c4 df ea ba
00c0   df 83 f5 a0 09 d0 00 3a 0c 19 b6 94 13 d6 64 fe
00d0   dc 14 bf 08 a3 b5 45 f3 90 e9 87 3d 22 20 e5 cf
00e0   ce ac e6 8d f2 75 8c ff d0 6b a0 46 13 eb 5e ce
00f0   3f 16 aa 82 dc 5b 42 52 59 ea c9 62 53 5c 1e c5
0100   ce d0 30 ca 58 08 1f 63 99 f6 b4 78 8b 2e 5e fb
0110   02 03 01 00 01 00 0d 02 00 00 00 00 00 00 00 00
0120   48 e9 f5 0b 00 21 0e 01 43 5b e2 8c ce 63 0c 13
0130   00 00 f1 69 2b b6 76 d7 f6 f9 63 53 6f 01 87 02
0140   e6 3b a3 e8 64 98 85 da d1 cd 57 d1 83 60 04 d1
0150   50 7d d4 ea 6e 5f b3 50 4f 09 f8 e6 e1 bc f4 ee
0160   3d fd c5 b6 a2 c5 11 2e d4 78 0c 2b 9f f2 63 6e
0170   b6 ca af 59 b3 d2 dd 55 ea c4 89 5b 3b 01 fb 65
0180   80 f3 15 91 dd 19 e9 80 ac 99 a4 f5 c2 74 c2 55
0190   68 8b d3 85 df 39 02 6e e2 09 d4 88 b6 fa c5 29
01a0   4e 87 6e 1d d6 50 00 47 51 9b a3 e6 ed af a0 ac
```

```
01b0   01 d4 3e 53 f0 c0 bc ef 90 33 b0 49 f7 c3 cc 25
01c0   5f c4 00 00 00 00
```

La opción ***CGA*** inicia en la posición 0x0056 y finaliza en la posición 0x0115 como se muestra a continuación:

```
0050   00 0c 29 c4 42 ba 0b 18 01 00 35 91 9b aa e5 04
0060   97 5b 9b e4 78 1b c8 41 6f 18 20 04 00 00 00 00
0070   00 00 00 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d
0080   01 01 01 05 00 03 81 8d 00 30 81 89 02 81 81 00
0090   c9 06 49 c1 7b 65 02 2e b0 08 d4 fa e4 82 d3 52
00a0   ee 8b 79 c0 d0 64 93 61 a2 84 3d 25 0b 32 9b 2b
00b0   de f0 a8 e8 11 55 ae 96 fb a2 35 0d c4 df ea ba
00c0   df 83 f5 a0 09 d0 00 3a 0c 19 b6 94 13 d6 64 fe
00d0   dc 14 bf 08 a3 b5 45 f3 90 e9 87 3d 22 20 e5 cf
00e0   ce ac e6 8d f2 75 8c ff d0 6b a0 46 13 eb 5e ce
00f0   3f 16 aa 82 dc 5b 42 52 59 ea c9 62 53 5c 1e c5
0100   ce d0 30 ca 58 08 1f 63 99 f6 b4 78 8b 2e 5e fb
0110   02 03 01 00 01 00
```

El campo correspondiente a los parámetros CGA (ver Figura 2.19) es de tamaño variable y su longitud depende del tamaño de la clave usada y en particular dicho campo está compuesto por el modificador final, el prefijo de red, la clave pública, entre otros (ver Apéndice C).

La opción **Timestamp** inicia en la posición 0x0116 y finaliza en la posición 0x0125 como se muestra a continuación:

```
0110   02 03 01 00 01 00 0d 02 00 00 00 00 00 00 00 00
0120   48 e9 f5 0b 00 21 0e 01 43 5b e2 8c ce 63 0c 13
```

El valor del campo *Timestamp* (ver Figura 2.23) es construido en base a la cantidad de segundos transcurridos desde la medianoche del 1 de enero de 1970, sin contar segundos intercalares.

Para que pueda haber una comunicación bidireccional, el reloj en los sistemas debe estar sincronizado. Para cada paquete saliente se genera un nuevo valor de Timestamp.

La opción **Nonce** inicia en la posición 0x0126 y finaliza en la posición 0x012d como se muestra a continuación:

```
0120   48 e9 f5 0b 00 21 0e 01 43 5b e2 8c ce 63 0c 13
```

El valor del campo *Nonce* (ver Figura 2.24) es generado aleatoriamente por el remitente del mensaje y almacenado internamente para verificar un eventual paquete en respuesta a dicha solicitud.

La opción RSASignature inicia en la posición 0x012e y finaliza en la posición 0x01c5 como se muestra a continuación.

```
0120   48 e9 f5 0b 00 21 0e 01 43 5b e2 8c ce 63 0c 13
0130   00 00 f1 69 2b b6 76 d7 f6 f9 63 53 6f 01 87 02
```

```
0140   e6 3b a3 e8 64 98 85 da d1 cd 57 d1 83 60 04 d1
0150   50 7d d4 ea 6e 5f b3 50 4f 09 f8 e6 e1 bc f4 ee
0160   3d fd c5 b6 a2 c5 11 2e d4 78 0c 2b 9f f2 63 6e
0170   b6 ca af 59 b3 d2 dd 55 ea c4 89 5b 3b 01 fb 65
0180   80 f3 15 91 dd 19 e9 80 ac 99 a4 f5 c2 74 c2 55
0190   68 8b d3 85 df 39 02 6e e2 09 d4 88 b6 fa c5 29
01a0   4e 87 6e 1d d6 50 00 47 51 9b a3 e6 ed af a0 ac
01b0   01 d4 3e 53 f0 c0 bc ef 90 33 b0 49 f7 c3 cc 25
01c0   5f c4 00 00 00 00
```

La clave privada utilizada para calcular la firma del campo Digital Signature (ver Figura 2.21) es obtenida durante la creación de los parámetros CGA y los datos utilizados para la generación de dicha firma se muestran a continuación:

SEND TAG definido en [8]:

```
08 6f ca 5e 10 b2 00 c9 9c 8c e0 01 64 27 7c 08
```

Dirección fuente:

```
20 04 00 00 00 00 00 00 34 cc 6b ba 59 47 d2 08
```

Dirección Destino:

```
ff 02 00 00 00 00 00 00 00 00 00 01 ff ba f4 4c
```

Campo *Type, Code y Checksum*. El valor del Checksum intermedio es calculado tomando en cuenta las opciones SEND agregadas hasta el momento (CGA, Timestamp y Nonce).

```
0036   87 00 85 5e 00 00 00 00 20 04 00 00 00 00 00 00
0046   38 69 5d db cd ba f4 4c
```

Opciones ND y SEND:

```
004e   01 01 00 0c 29 c4 42 ba 0b 18 01 00 35 91 9b aa
005e   e5 04 97 5b 9b e4 78 1b c8 41 6f 18 20 04 00 00
006e   00 00 00 00 00 30 81 9f 30 0d 06 09 2a 86 48 86
007e   f7 0d 01 01 01 05 00 03 81 8d 00 30 81 89 02 81
008e   81 00 c9 06 49 c1 7b 65 02 2e b0 08 d4 fa e4 82
009e   d3 52 ee 8b 79 c0 d0 64 93 61 a2 84 3d 25 0b 32
00ae   9b 2b de f0 a8 e8 11 55 ae 96 fb a2 35 0d c4 df
00be   ea ba df 83 f5 a0 09 d0 00 3a 0c 19 b6 94 13 d6
00ce   64 fe dc 14 bf 08 a3 b5 45 f3 90 e9 87 3d 22 20
00de   e5 cf ce ac e6 8d f2 75 8c ff d0 6b a0 46 13 eb
00ee   5e ce 3f 16 aa 82 dc 5b 42 52 59 ea c9 62 53 5c
00fe   1e c5 ce d0 30 ca 58 08 1f 63 99 f6 b4 78 8b 2e
010e   5e fb 02 03 01 00 01 00 0d 02 00 00 00 00 00 00
011e   00 00 48 e9 f5 0b 00 21 0e 01 43 5b e2 8c ce 63
```

La firma generada resultante es:

```
a3 e8 64 98 85 da d1 cd 57 d1 83 60 04 d1 50 7d
d4 ea 6e 5f b3 50 4f 09 f8 e6 e1 bc f4 ee 3d fd
c5 b6 a2 c5 11 2e d4 78 0c 2b 9f f2 63 6e b6 ca
```

```
af 59 b3 d2 dd 55 ea c4 89 5b 3b 01 fb 65 80 f3
15 91 dd 19 e9 80 ac 99 a4 f5 c2 74 c2 55 68 8b
d3 85 df 39 02 6e e2 09 d4 88 b6 fa c5 29 4e 87
6e 1d d6 50 00 47 51 9b a3 e6 ed af a0 ac 01 d4
3e 53 f0 c0 bc ef 90 33 b0 49 f7 c3 cc 25 5f c4
```

Finalmente se agrega la opción RSASignature al mensaje NS y se conforma el paquete ICMPv6 calculando el valor final del campo Payload y Checksum.

B. Analizando un mensaje NA entrante

A continuación se muestra el flujo de bytes asociado al paquete ICMPv6 N°2 del caso de prueba 1.

Dirección Fuente: `2004::3869:5ddb:cdba:f44c`
Dirección Destino: `2004::34cc:6bba:5947:d208`

```
0000   00 0c 29 c4 42 ba 00 0c 29 c4 87 be 86 dd 60 00
0010   00 00 01 90 3a ff 20 04 00 00 00 00 00 00 38 69
0020   5d db cd ba f4 4c 20 04 00 00 00 00 00 00 34 cc
0030   6b ba 59 47 d2 08 88 00 8d 16 e0 00 00 00 20 04
0040   00 00 00 00 00 00 38 69 5d db cd ba f4 4c 02 01
0050   00 0c 29 c4 87 be 0b 18 01 00 1b be bd c4 5f 26
0060   a4 c0 66 6b eb 46 73 44 fe 32 20 04 00 00 00 00
0070   00 00 00 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d
0080   01 01 01 05 00 03 81 8d 00 30 81 89 02 81 81 00
0090   b0 89 3b 51 88 d3 f2 3c 9b a0 98 4f 99 01 b4 5f
00a0   de 81 89 b3 d6 ba 96 ff a7 05 b3 49 4c 28 ce 60
00b0   2e ba e7 f6 76 f4 c2 87 f9 51 36 be 08 12 ab c2
00c0   e5 00 50 8a b5 be fb 04 79 99 79 b4 ee 88 5b 45
00d0   b7 f1 88 a9 e2 e4 52 3a 75 9b ea 94 87 38 56 49
00e0   8d 63 66 b6 78 1e d0 41 4a f5 25 e9 cf 69 50 9b
00f0   f2 b9 68 91 a9 8c e6 8a 6f 61 8f 45 8d 40 6a fc
0100   a4 26 ae 1e a8 7d 8b 48 88 c2 20 b3 c3 e3 93 2b
0110   02 03 01 00 01 00 0d 02 00 00 00 00 00 00 00 00
0120   48 e9 f5 28 00 00 0e 01 43 5b e2 8c ce 63 0c 13
0130   00 00 9c 96 4f af a4 c3 41 89 ac 10 f6 26 24 c5
0140   62 35 0f 5e d0 99 38 d0 c8 62 1c 73 e7 47 ad 2e
0150   ab 93 b7 30 f1 06 ec 3e 72 b9 96 b3 a8 11 fc d1
0160   a2 0f 5e 86 85 a4 df ed 41 22 67 eb a4 cb 8f 7e
0170   9d de 72 b2 d0 5d df 09 db 54 2a e5 97 4c c1 df
0180   76 b8 29 cc a9 a8 0a 1e 1d fb 31 fc 96 73 d3 6b
0190   c5 d4 95 25 00 2c dd 06 1b f3 4e 81 28 c8 04 b8
01a0   de e5 1c a0 bc a7 d1 e5 09 71 ed 90 06 77 3d f1
01b0   00 73 70 3b 5d 82 10 96 57 1d 14 aa 8f b9 44 d3
01c0   8e 04 00 00 00 00
```

Una vez recibido el paquete se procede a revisar las opciones SEND iniciando la verificación de la opción CGA como se muestra a continuación:

Se verifica que la dirección 2004:0:0:0:3869:5ddb:cdba:f44c es efectivamente CGA, tomando como datos de entrada los parámetros CGA del paquete entrante como se muestra a continuación:

```
0050   00 0c 29 c4 87 be 0b 18 01 00 1b be bd c4 5f 26
0060   a4 c0 66 6b eb 46 73 44 fe 32 20 04 00 00 00 00
0070   00 00 00 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d
0080   01 01 01 05 00 03 81 8d 00 30 81 89 02 81 81 00
0090   b0 89 3b 51 88 d3 f2 3c 9b a0 98 4f 99 01 b4 5f
00a0   de 81 89 b3 d6 ba 96 ff a7 05 b3 49 4c 28 ce 60
00b0   2e ba e7 f6 76 f4 c2 87 f9 51 36 be 08 12 ab c2
00c0   e5 00 50 8a b5 be fb 04 79 99 79 b4 ee 88 5b 45
00d0   b7 f1 88 a9 e2 e4 52 3a 75 9b ea 94 87 38 56 49
00e0   8d 63 66 b6 78 1e d0 41 4a f5 25 e9 cf 69 50 9b
00f0   f2 b9 68 91 a9 8c e6 8a 6f 61 8f 45 8d 40 6a fc
0100   a4 26 ae 1e a8 7d 8b 48 88 c2 20 b3 c3 e3 93 2b
0110   02 03 01 00 01 00
```

El valor del resumen *Hash1* calculado es 3a 69 5d db cd ba f4 4c, mientras que el *Hash2* calculado es 00 00 1a 41 15 a2 36 32 23 28 13 23 2c 49.

Al comparar el valor del *Hash1* con el identificador de la interfaz que viene en la dirección fuente e ignorando los bits *Sec*, "u" y "g", se obtiene que son iguales; seguidamente se verifica que el *Hash2* tenga los 16**Sec* primeros bits igual a cero.

Luego, se procede a verificar las opciones Timestamp y Nonce en conformidad con lo descrito en el Marco Teórico (ver Capítulo 2). El valor del campo *Nonce* de este mensaje debe coincidir con el valor del campo *Nonce* de la solicitud.

Finalmente, se procede a verificar la firma digital usando la clave pública del remitente contenido en la estructura de parámetros CGA (ver Figura 2.18) como se muestra a continuación:

```
0050   00 0c 29 c4 87 be 0b 18 01 00 1b be bd c4 5f 26
0060   a4 c0 66 6b eb 46 73 44 fe 32 20 04 00 00 00 00
0070   00 00 00 30 81 9f 30 0d 06 09 2a 86 48 86 f7 0d
0080   01 01 01 05 00 03 81 8d 00 30 81 89 02 81 81 00
0090   b0 89 3b 51 88 d3 f2 3c 9b a0 98 4f 99 01 b4 5f
00a0   de 81 89 b3 d6 ba 96 ff a7 05 b3 49 4c 28 ce 60
00b0   2e ba e7 f6 76 f4 c2 87 f9 51 36 be 08 12 ab c2
00c0   e5 00 50 8a b5 be fb 04 79 99 79 b4 ee 88 5b 45
00d0   b7 f1 88 a9 e2 e4 52 3a 75 9b ea 94 87 38 56 49
00e0   8d 63 66 b6 78 1e d0 41 4a f5 25 e9 cf 69 50 9b
00f0   f2 b9 68 91 a9 8c e6 8a 6f 61 8f 45 8d 40 6a fc
0100   a4 26 ae 1e a8 7d 8b 48 88 c2 20 b3 c3 e3 93 2b
0110   02 03 01 00 01 00
```

Los datos usados para verificar la firma son los siguientes:

SEND TAG definido en [8]:

```
08 6f ca 5e 10 b2 00 c9 9c 8c e0 01 64 27 7c 08
```

Dirección fuente:

```
20 04 00 00 00 00 00 00 38 69 5d db cd ba f4 4c
```

Dirección Destino:

```
20 04 00 00 00 00 00 00 34 cc 6b ba 59 47 d2 08
```

Campo Type, Code y Checksum. El valor del Checksum es calculado tomando en cuenta las opciones SEND verificadas que preceden a la opción RSASignature.

```
0036   88 00 c9 b6 e0 00 00 00 20 04 00 00 00 00 00 00
0046   38 69 5d db cd ba f4 4c
```

Opciones ND y SEND:

```
004e   02 01 00 0c 29 c4 87 be 0b 18 01 00 1b be bd c4
005e   5f 26 a4 c0 66 6b eb 46 73 44 fe 32 20 04 00 00
006e   00 00 00 00 00 30 81 9f 30 0d 06 09 2a 86 48 86
007e   f7 0d 01 01 01 05 00 03 81 8d 00 30 81 89 02 81
008e   81 00 b0 89 3b 51 88 d3 f2 3c 9b a0 98 4f 99 01
009e   b4 5f de 81 89 b3 d6 ba 96 ff a7 05 b3 49 4c 28
00ae   ce 60 2e ba e7 f6 76 f4 c2 87 f9 51 36 be 08 12
00be   ab c2 e5 00 50 8a b5 be fb 04 79 99 79 b4 ee 88
00ce   5b 45 b7 f1 88 a9 e2 e4 52 3a 75 9b ea 94 87 38
00de   56 49 8d 63 66 b6 78 1e d0 41 4a f5 25 e9 cf 69
00ee   50 9b f2 b9 68 91 a9 8c e6 8a 6f 61 8f 45 8d 40
00fe   6a fc a4 26 ae 1e a8 7d 8b 48 88 c2 20 b3 c3 e3
010e   93 2b 02 03 01 00 01 00 0d 02 00 00 00 00 00 00
011e   00 00 48 e9 f5 28 00 00 0e 01 43 5b e2 8c ce 63
```

La firma recibida es:

```
0120   48 e9 f5 28 00 00 0e 01 43 5b e2 8c ce 63 0c 13
0130   00 00 9c 96 4f af a4 c3 41 89 ac 10 f6 26 24 c5
0140   62 35 0f 5e d0 99 38 d0 c8 62 1c 73 e7 47 ad 2e
0150   ab 93 b7 30 f1 06 ec 3e 72 b9 96 b3 a8 11 fc d1
0160   a2 0f 5e 86 85 a4 df ed 41 22 67 eb a4 cb 8f 7e
0170   9d de 72 b2 d0 5d df 09 db 54 2a e5 97 4c c1 df
0180   76 b8 29 cc a9 a8 0a 1e 1d fb 31 fc 96 73 d3 6b
0190   c5 d4 95 25 00 2c dd 06 1b f3 4e 81 28 c8 04 b8
01a0   de e5 1c a0 bc a7 d1 e5 09 71 ed 90 06 77 3d f1
01b0   00 73 70 3b 5d 82 10 96 57 1d 14 aa 8f b9 44 d3
01c0   8e 04 00 00 00 00
```

Si la verificación es exitosa se procede a remover las opciones SEND incluidas en el mensaje NA, se calcula el Payload y el Checksum final para luego reinyectar el paquete.

Si la verificación no fue exitosa, se procede a desechar el paquete.

Referencias Bibliográficas

[1] **T. Narten, E. Nordmark, W. Simpson and H. Soliman**, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 4861, September 2007.

[2] **S. Thomson, T. Narten and T Jinmei**, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[3] **S. Deering and R. Hinden**, "Internet Protocol Version 6 (IPv6) Specification", RFC 2460, December 1998.

[4] **A. Conta, S. Deering and M. Gupta**, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, March 2006.

[5] **D. Johnson, C. Perkins and J. Arkko**, "Mobility Support in IPv6", RFC 3775, June 2004.

[6] **R. Draves and D. Thaler**, "Default Router Preferences and More-Specific Routes", RFC 4191, November 2005.

[7] **P. Nikander, J. Kempf and E. Nordmark**, "IPv6 Neighbor Discovery (ND) Trust Models and Threats", RFC 3756, May 2004.

[8] **J. Arkko, J. Kempf, B. Zill and P. Nikander**, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[9] **S. Hagen**, "IPv6 Essentials", O'Reilly, Second Edition, May 2006.

[10] **T. Aura**, "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[11] **R. Hinden and S. Deering**, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[12] **J. Arkko and M. Bagnulo**, "Cryptographically Generated Addresses (CGA) Extension Field Format", RFC 4581, October 2006.

[13] **R. Housley, W. Polk**, W. Ford and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[14] **J. Davies**, "Understanding IPv6", Microsoft Press, Second Edition, January 2008.

[15] **I. Sommerville**, "Ingeniería del Software", Pearson, Séptima Edición, 2005.

Glosario de Términos

ADD (Authorization Delagtion Discovery): proceso a través del cual los nodos SEND son autorizados por un ancla de confianza. Los host son configurados con un camino de certificación y los router con un certificado de autorización.

API (Application Programming Interface): conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

ARP (Address Resolution Protocol): protocolo del nivel de red responsable de encontrar la dirección hardware (Ethernet MAC) que corresponde a una determinada dirección IP.

ASN.1 (Abstract Syntax Notation One): lenguaje formal utilizado para describir mensajes que pueden ser intercambiados entre gran cantidad de aplicaciones. En SNMP se utiliza ASN.1 para la representación de los objetos administrados.

CA (Certification Authority): entidad de confianza responsable de emitir y revocar los certificados digitales utilizados en la firma electrónica, para lo cual se emplea la criptografía de clave pública.

Certificado Digital: documento digital mediante el cual un tercero confiable (una CA) garantiza la vinculación entre la identidad de un sujeto o entidad y su clave pública.

CGA (Cryptographically Generated Addresses): método a través del cual una dirección IPv6 es generada por una función hash, a partir de una clave pública y algunos parámetros auxiliares.

Checksum: suma de verificación o Checksum es una forma de control de error muy simple para proteger la integridad de datos, comprobando que no hayan sido corrompidos.

CPA (Certification Path Advertisement): mensaje de anuncio usado en el proceso ADD.

CPS (Certification Path Solicitation): mensaje de solicitud usado en el proceso ADD.

CRL (Certification Revocation List): lista firmada digitalmente por una CA que contiene los certificados revocados junto con su fecha y razón de revocación.

DER (Distinguish Encoding Rules): representa uno de los formatos de codificación definido como parte del estándar ASN.1.

DoS (Denial of Services): ataque informático. El atacante provoca que un conjunto de computadores queden fuera de servicio y sean inaccesibles a los usuarios legítimos.

FQDN (Fully Qualified Domain Name): identificador que incluye el nombre del computador y el nombre de dominio asociado a ese equipo.

ICMP (Internet Control Message Protocol): protocolo utilizado para el control y notificación de errores en el Protocolo Internet.

ICMPv6 (Internet Control Message Protocol version 6): versión 6 del protocolo ICMP.

IP (Internet Protocol): protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

IPv4 (Internet Protocol version 4): representa la versión 4 del Protocolo de Internet. Es el protocolo de capa de red ampliamente utilizado y que en la actualidad está siendo reemplazado por su sucesor IPv6, debido al agotamiento de direcciones en IPv4.

IPv6 (Internet Protocol version 6): nuevo protocolo perteneciente a la capa de red, diseñado para reemplazar el Protocolo de Internet (IPv4) con la finalidad de solventar los problemas de direccionamiento que existen en la actualidad.

MAC (Medium Access Control): identificador de 48 bits (6 bytes) que corresponde de forma única a una tarjeta o interfaz de red.

MiM (Man in the Middle): ataque informático. El atacante adquiere la capacidad de leer, insertar y modificar a voluntad los mensajes intercambiados entre dos host, sin que ninguno de ellos conozca que el enlace ha sido entre ellos ha sido violado.

MTU (Maximum Transmission Unit): expresa el tamaño en bytes del datagrama más grande que puede pasar por una capa de un protocolo de comunicaciones.

NTP (Network Time Protocol): protocolo de internet utilizado para sincronizar los relojes de los sistemas computacionales a través del ruteo de paquetes en redes con latencia variable.

Paquete: conjunto estructurado de bytes que forma la unidad básica de comunicación del protocolo IP (en todas sus versiones).

Ping: herramienta que permite probar la conectividad entre dos hosts. Se basa en el envío de mensajes ICMP de Echo Request para conocer si un dispositivo se encuentra activo.

Protocolo: conjunto de reglas que establece cómo debe realizarse una comunicación.

RFC (Request For Comments): documento de especificaciones que se expone públicamente para su discusión.

Router: dispositivo físico o computador que permite asegurar el enrutamiento de paquetes entre redes o determinar la ruta que debe tomar el paquete de datos.

SHA-1 (Secure Hash Algorithm-1): algoritmo que produce un resumen (hash) de 160 bits de un mensaje que puede tener un tamaño máximo de 2^{64} bits.
Ubuntu: distribución GNU/Linux que ofrece un sistema operativo basado en Debian.

UTC (Universal Time Clock): estándar de tiempo internacional.

VMware Workstation: software que proporciona un sistema de virtualización para Windows, Linux, y Mac OS X.

X.509: estándar UIT-T para infraestructuras de claves públicas que especifica, entre otras cosas, un formato estándar para certificados de claves públicas y un algoritmo de validación de la ruta de certificación.

Wireshark: herramienta multiplataforma de análisis de red.