Redes emergentes

PID_00195521

Joan Borrell Viader
Helena Rifà Pous
Sergi Robles Martínez

Universitat Oberta de Catalunya

Índice

Introducción

En este módulo didáctico presentamos algunos de los principales problemas de seguridad relacionados con las nuevas redes que están surgiendo en los últimos años. Se trata principalmente de redes inalámbricas, distribuidas e incluso autoorganizadas, que están formadas por dispositivos móviles y con limitaciones de recursos mucho más estrictas que las computadoras que se utilizan en las redes tradicionales.

Comenzamos el primer apartado explicando qué entendemos por redes emergentes y cuáles son las nuevas vulnerabilidades de seguridad que plantean.

A continuación, en el segundo apartado, nos centramos en las redes *ad hoc* y describimos los mecanismos utilizados para proteger las operaciones de construcción de la red. Vemos cómo se puede implementar un proceso de descubrimiento de vecinos que garantice su identidad y cómo se puede establecer la información de encaminamiento (tablas de encaminamiento o cachés) correcta para enviar paquetes a través de la red. Finalmente, en este apartado describimos también cuáles son los riesgos en la privacidad y cómo podemos minimizarlos.

En el tercer apartado estudiamos las características de las redes de sensores y vemos cómo se pueden resolver dos de sus principales problemas de seguridad: la gestión de claves y la agregación de datos. Las limitaciones de los sensores hacen que todas las soluciones tengan que basarse en operaciones criptográficas muy ligeras, como funciones resumen (*hash*).

Acabamos el módulo presentando los protocolos tolerantes a retardos e interrupciones (*delay and disruption tolerant networking*, DTN), destinados a las situaciones en las que o bien no existe un canal continuo entre las partes que se comunican, o bien los retardos en las comunicaciones son muy elevados. En el enfoque DTN, las comunicaciones se llevan a cabo exclusivamente en el nivel de aplicación, entre nodos adyacentes cuando estos tiene la oportunidad de comunicarse, y sin necesidad de conectividad continua de extremo a extremo. Este método permite las comunicaciones incluso en las situaciones más adversas, pero a costa de perder la interactividad, ya que cada paso de una comunicación puede llevar un tiempo no limitado. Los principales problemas de seguridad que plantean los protocolos DTN son la gestión de las claves criptográficas y el encaminamiento de la información.

Objetivos

Los materiales didácticos de este módulo deben permitir al alumnado alcanzar los siguientes objetivos:

1. Entender los problemas de seguridad de las redes emergentes.

2. Comprender las características de los algoritmos de seguridad que se utilizan en entornos en los que los costes computacionales, de energía y de transmisión de datos a través de la red son realmente importantes.

3. Conocer el funcionamiento de diversos mecanismos y protocolos de seguridad diseñados para redes *ad hoc*, de sensores y protocolos tolerantes a retardos e interrupciones.

1. Redes emergentes

En este apartado resumiremos las características básicas de las redes emergentes y describiremos cuáles son las vulnerabilidades a las que son susceptibles dada su naturaleza básica de funciotambién se denominannamiento.

1.1. Descripción de las redes

La infraestructura de una red tradicional, es decir, el conjunto de elementos que conforma la red, excepto los terminales finales, pertenece a una empresa u operadora que da servicio de conexión a sus usuarios. Los usuarios que utilizan la red confían en la operadora y asumen que esta les proporcionará un buen servicio de transmisión y recepción de paquetes.

En los últimos años, la tecnología inalámbrica ha bajado mucho de precio, lo que ha facilitado el desarrollo de sistemas basados en esta tecnología y la adquisición de equipamiento por parte de los usuarios. El incremento de terminales inalámbricos ha hecho surgir las redes inalámbricas basadas en una topología en malla[1]. En una topología de este tipo, cada nodo está conectado a uno de los otros nodos o a más de uno. La infraestructura de dichas redes puede ser descentralizada, sin servidor central ni el soporte de una operadora, o centralizada.

(1) En inglés, *wireless mesh networks* (WMN).

En el caso de redes WMN descentralizadas, los nodos utilizan la implementación del modo *ad hoc* del estándar IEEE 802.11. Por este motivo, también se denominan **redes *ad hoc***.

En las redes *ad hoc*, la comunicación se basa en la cooperación de un gran número de dispositivos individuales inalámbricos que permiten que un mensaje acabe llegando, salto a salto, de un punto a otro de la red. Cada usuario debe tener unas capacidades de encaminamiento: una especie de carrera de relevos en la que la densidad de usuarios consigue que el relevo represente un esfuerzo menor.

MANET

Las MANET (*mobile ad-hoc networks*) son redes *ad hoc* generadas por la autoconfiguración de nodos encaminadores que forman una topología arbitraria. Dado que los nodos se pueden mover libremente, se considera que la topología de red cambia muy rápido y de forma impredecible.

Las redes WMN centralizadas corresponden a una mezcla de topologías *ad hoc* e infraestructura (o modo con punto de acceso). Se trata de redes con infraestructura que, además, permiten la unión de dispositivos que están fuera del rango de cobertura de los puntos de acceso, pero dentro del rango de cobertura

(2) En inglés, *transit access points* (TAP).

de algún punto de acceso de tráfico[2]. Los TAP son nodos especiales controlados por la operadora que permiten la retransmisión de los paquetes de los clientes más allá de su radio de cobertura.

Otro de los tipos de redes que se han extendido en los últimos años son las redes de sensores[3]. Se trata de redes inalámbricas formadas por dispositivos autónomos que pueden trabajar de manera cooperativa para monitorizar las condiciones físicas o ambientales (por ejemplo, temperatura, sonido, vibración, presión, movimiento, polución) de una zona.

(3)En inglés, *wireless sensor network* (WSN).

Las **redes de sensores** se pueden considerar un subconjunto muy especializado de las WMN, en el que los nodos tienen recursos muy limitados (son pequeños nodos sensores) y están orientados a una tarea muy específica de monitorización.

En la literatura de las WSN se utiliza, por lo general, una nomenclatura diferente de la utilizada en WMN, en la que los TAP se denominan estaciones base o supernodos. Por tanto, las estaciones base son nodos de la WSN con capacidad computacional ampliada y más recursos en lo que concierne a batería y comunicaciones.

Los protocolos DTN[4] han surgido hace poco para dar una solución de comunicación en situaciones en las que o bien no se puede garantizar una conectividad de extremo a extremo, o bien los retardos en las comunicaciones son tan elevados que los protocolos actuales de Internet no son aplicables.

(4)Del inglés *delay and disruption tolerant network*.

Las situaciones en las que se aplican los protocolos DTN son muy variadas, desde comunicaciones interplanetarias hasta redes WSN (o MANET) con conectividad intermitente o nula, ya sea por la elevada movilidad de sus nodos, por la inexistencia de una infraestructura de comunicación o por la excesiva distancia que separa sus nodos.

Temporizadores

Recordemos que el protocolo TCP tiene varios temporizadores para controlar la transmisión de la información, con valores máximos de pocos minutos, y que asume una tasa de error muy baja.

1.2. Vulnerabilidades

Las características más relevantes de las redes emergentes son que se basan en tecnologías de comunicación inalámbricas (normalmente radio, aunque también se considera el uso de infrarrojos o ultrasonidos) y que la cooperación de los usuarios es uno de los puntos clave para hacerlas atractivas y viables.

Las redes inalámbricas facilitan al atacante el acceso al medio y, por tanto, son especialmente vulnerables a lo siguiente:

- **Escuchas no autorizadas (*eavesdropping*)**. Colocando una antena de recepción en el lugar adecuado, un atacante puede oír la información que se envía o recibe por el medio radio. El ataque de escuchas no autorizadas se clasifica en la categoría de ataques pasivos, que consisten en escuchar el medio y analizar los datos capturados sin interactuar con él. Los ataques pasivos se pueden evitar, y así se suele hacer, cifrando la información intercambiada.

- **Alteración de los datos**. Se trata de un ataque activo en el que una entidad maliciosa modifica el contenido de los mensajes que se intercambian dos entidades o más. Normalmente, estos ataques se llevan a cabo mediante el ataque del hombre a medio camino *(man-in-the-middle*, MITM), en el que el atacante se convierte en una passarela entre dos entidades y, por tanto, tiene control absoluto sobre el tráfico entre ellas.

- **Ataques de suplantación de identidad (*spoofing*)**. Un atacante se hace pasar por otro nodo para tener acceso a zonas restringidas o limitadas a las que solo puede acceder un nodo autorizado.

- **Ataques por múltiples falsas identidades (*sybil*)**. En este caso, un atacante adopta de forma deliberada múltiples identidades legítimas de la red (roba los datos de otros usuarios o genera datos nuevos) con el objetivo de tener una gran influencia en la red o grupo, por ejemplo en servicios en los que se utilizan votaciones, reputaciones, etc.

- **Negación del servicio (*DoS*)**. Este tipo de ataque tiene lugar en el ámbito físico, cuando un nodo malicioso envía de forma indiscriminada mensajes que consumen el ancho de banda disponible de la red, y consigue la indisponibilidad temporal o permanente de un servicio. En redes inalámbricas se dan básicamente dos tipos de ataques DoS:
  - **Bombas electrónicas**, en las que el atacante genera una señal de gran potencia por el mismo medio de comunicación, de modo que impide que se produzca el intercambio de mensajes. Este tipo de ataque es difícil de evitar, aunque se puede paliar un poco por medio de técnicas de espectro ensanchado[5] y salto de frecuencias[6].

  - **Inundación de datos**, donde los nodos comprometidos envían grandes cantidades de información a otros nodos para saturarlos. En este caso, los ataques se pueden evitar mediante sistemas de detección de intrusos[7] y el posterior control de acceso que inhabilite (deje fuera del grupo seguro) el nodo que inunda/ataca.

Por otro lado, el hecho de que los nodos de las redes inalámbricas puedan ser dispositivos pequeños y móviles también introduce ciertas vulnerabilidades en el sistema:

(5)En inglés, *spread spectrum*.

(6)En inglés, *frequency hopping*.

(7)En inglés, *intrussion detection system* (IDS).

- **Ataques físicos**. Los usuarios maliciosos tienen acceso físico a los nodos de la red y pueden llevar a cabo ataques, como por ejemplo destruir nodos, capturarlos y sustraerles información sensible o comprometer las claves criptográficas para transmitir información falsa a la red.

- **Localización**. En muchos casos, para garantizar la movilidad de un dispositivo, es preciso trazar su ubicación, lo cual se puede considerar un ataque a la privacidad.

- **Limitación de recursos**. Generalmente, los dispositivos móviles son pequeños y, por tanto, con recursos más limitados de potencia, almacenamiento y batería. De las tres limitaciones, la de batería es la más representativa, ya que su progreso tecnológico es mucho más lento que el de procesadores y memorias. Normalmente, las limitaciones de batería llevan a reducir el número de operaciones computacionales que realiza el dispositivo inalámbrico, lo que suele derivar en una implementación de protocolos de seguridad pobre.

Y finalmente, las redes que se basan en la cooperación de sus nodos se pueden ver perjudicadas por la existencia de nodos no colaboradores. Los nodos que no colaboran correctamente con la red con el objetivo de dañarla o perjudicar a sus integrantes son nodos maliciosos. Los que no colaboran para ahorrar recursos y al mismo tiempo utilizan los servicios de la red a expensas de los demás usuarios son nodos egoístas.

Maliciosos y egoístas

Utilizando la terminología de nodos maliciosos y egoístas, podríamos indicar que un diseñador de virus es malicioso, mientras que un emisor de correo basura (*spammer*) es egoísta.

Los nodos maliciosos y egoístas pueden atacar a los protocolos colaborativos de la red, en particular a los protocolos de encaminamiento. En redes descentralizadas en las que todos los usuarios colaboran en el encaminamiento de paquetes, los ataques al encaminamiento degradan el servicio de forma indirecta por inhabilitación de las comunicaciones:

- **Disrupción de rutas**. Un atacante evita que se descubra una ruta entre dos nodos. El objetivo es degradar la calidad de servicio de la red.

- **Desviación de rutas**. En este caso, el adversario no evita que se cree una ruta, pero consigue que esta se establezca de forma diferente a lo que marca el protocolo (no se crean las rutas más cortas, rápidas, fiables, etc.). El objetivo es incrementar el control del adversario sobre algunas víctimas, y poder escuchar o incluso modificar los paquetes de datos que envía.

Existen diferentes formas de implementar tales ataques, pero podemos destacar las siguientes:

- **Ataques *blackhole/sinkhole***. En este caso, un nodo atacante se presenta como una buena opción para el encaminamiento y se convierte en un nodo atractivo por el que pasa gran parte de los paquetes de la red. Una vez que el nodo ha conseguido atraer el tráfico de la red, elimina los paquetes,

con lo que provoca una disrupción de rutas (ataque *blackhole*) o utiliza el tráfico para fines maliciosos (ataque *sinkhole*).

- **Ataques *warmhole***. En este caso, un nodo atacante crea un túnel oculto entre dos partes diferentes de una red descentralizada al que encamina ciertos mensajes de control de una zona a otra con el objetivo de distorsionar los mecanismos de encaminamiento.

- **Ataques al control de errores**. En este caso, un atacante envía mensajes de error falsos que invalidan el establecimiento correcto de una ruta.

- **Ataques de reinyección de paquetes**. En este caso, un nodo envía paquetes de encaminamiento antiguos con información que en su momento era correcta pero que ahora ya ha quedado desfasada y, por tanto, es inválida.

- **Ataques de generación y modificación de paquetes**. En este caso, un nodo crea o modifica paquetes con información de encaminamiento falsa.

En los siguientes apartados veremos cómo afectan estas vulnerabilidades al buen funcionamiento de las redes *ad hoc* y de sensores.

Para el caso de los protocolos DTN, aparte de las vulnerabilidades comentadas, hay que tener en cuenta otras especificidades, como por ejemplo el hecho de que pierde sentido cualquier infraestructura centralizada para la gestión de las claves criptográficas, dada la imposibilidad de interactuar de manera continua con el centro de la infraestructura. En relación con lo expuesto, cualquier protocolo clásico basado en una negociación de varios pasos[8] deja de poder utilizarse en estos entornos.

[8] Como el protocolo TLS (11).

Ved también

Los problemas de los protocolos DTN se estudian en el apartado 5 de este módulo.

2. Redes *ad hoc*

La característica más relevante de las redes *ad hoc* es el hecho de que los nodos de la red desempeñan dos papeles: actuar como terminales finales y llevar a cabo las funciones de encaminamiento.

Puesto que los nodos son terminales personales, los usuarios pueden manipular su funcionamiento para que actúen solo en busca del beneficio del propietario y perjudicando, si se da el caso, al resto de la red. Por otro lado, dado que se trata de terminales inalámbricos, móviles y relativamente pequeños, todos los nodos son vulnerables a ataques de usuarios maliciosos.

En este apartado estudiaremos los problemas de seguridad de las redes *ad hoc* en las operaciones que les son más propias y características, es decir, en las operaciones de gestión del encaminamiento de la red a partir de terminales finales. En primer lugar, veremos cómo podemos asegurar de forma eficiente que los vecinos de un nodo son quienes dicen ser, después analizaremos el funcionamiento de los protocolos de encaminamiento y finalmente veremos cuáles son los riesgos de privacidad de estas redes y cómo podemos minimizarlos.

2.1. Descubrimiento de los nodos vecinos

Uno de los principales problemas de trabajar con redes dinámicas en las que los nodos cambian eventualmente de estado (activos/inactivos), posición y condiciones de entorno es la complejidad que tienen los nodos origen para encontrar los nodos o servicios destino a los que quieren conectarse.

El primer paso para construir rutas entre nodos remotos es conocer cuáles son los terminales vecinos, es decir, aquellos con los que existe conexión directa.

El descubrimiento de vecinos se puede hacer a través de protocolos muy simples en los que un nodo envía una petición de descubrimiento en modo difusión (*broadcast*) y todos los nodos que reciben este paquete responden. Las respuestas que recibe el nodo emisor corresponden a los nodos que están en su rango de cobertura y, por tanto, con los que hay conexión directa.

Nota

Los protocolos de descubrimiento de vecinos se suelen denominar *hello protocols* y los mensajes enviados son *hello messages*.

Uno de los requisitos deseables de los protocolos de descubrimiento de vecinos es su capacidad de enviar mensajes en difusión que puedan ser autenticados. Es decir, un emisor debe ser capaz de enviar un mensaje y todos los receptores tienen que poder ser capaces de verificar su autenticidad, al mismo tiempo que hay que impedir que estos puedan utilizar la información recibida para efectuar ataques.

La solución tradicional de utilizar firmas digitales para la autenticación de mensajes no es aplicable en estos entornos, dado el gran volumen de mensajes *hello* que se gestiona en una red *ad hoc*, y las características limitadas de los dispositivos que suelen formar parte de dichas redes (dispositivos empotrados, móviles, etc.). Por este motivo, es preciso encontrar alternativas que no se basen en operaciones criptográficas complejas como las de clave pública.

Una de las propuestas más destacadas para la autenticación de una secuencia de mensajes en difusión es el protocolo TESLA. La robustez de TESLA se basa en dos elementos:

Lectura complementaria

El protocolo TESLA está definido en una RFC:

A. Perrig; D. Song; R. Canetti; J. D. Tygar; B. Briscoe (2005). "Timed efficient stream loss-tolerant authentication (TESLA): Multicast source authentication transform introduction". *RFC 4082* (*informational*) (junio).

1) Cadenas resumen (*hash*). Una cadena resumen es una secuencia de valores tal que cada elemento es el resultado de aplicar una función resumen sobre el elemento inmediatamente anterior. Una de sus principales propiedades es la unidireccionalidad, ya que sus elementos se pueden calcular muy fácilmente en una dirección, pero no en la dirección contraria.

2) Firmas MAC. Una firma MAC tiene dos entradas –un mensaje y una clave secreta– y produce una salida que permite verificar la integridad y la autenticidad del mensaje. Cualquier cambio en el mensaje o en la clave secreta desembocará en la generación de un resultado diferente.

Construcción y uso de las cadenas resumen

Una cadena resumen se construye de atrás hacia delante. Al principio se determina la longitud N de la cadena y se elige un valor aleatorio, que denominaremos v_N. Entonces, se puede calcular el resto de los valores de la cadena aplicando una función resumen de manera recursiva: $v_i = h(v_{i+1}) = h^{(N-i)}(v_N)$.

El uso de los valores de la cadena resumen se hace en orden creciente, desde v_0 hasta v_N (en el orden contrario al que se han generado los valores). Fijaos en que, dado v_i, es computacionalmente eficiente calcular $v_j = h^{(i-j)}(v_i)$ para cualquier $j > i$, pero no es posible calcular v_k con $k > i$. Así pues, el uso y la revelación de un valor v_i de la cadena no desvela ninguna información sobre los siguientes valores de la cadena.

Ejemplo

Uno de los primeros protocolos que utiliza las cadenas resumen es el esquema de autenticación con contraseña de Lamport. Consideremos un escenario en el que un usuario quiere "logarse" en un ordenador remoto. Para hacerlo, tiene que enviar la constraseña

a través de un canal inseguro. Para impedir que un intruso intercepte su contraseña y pueda utilizarla más tarde, el usuario dispone de un juego de claves $\{x_0, x_1,\ldots,X_{1000}\}$ almacenado en el equipo; cada vez que utiliza una contraseña, esta se convierte en inválida para ser utilizada de nuevo.

Para minimizar los requisitos de almacenamiento del usuario y del equipo remoto, se propone que el valor x_{1000} sea escogido por el usuario, y que los demás valores se definan como $x_i = f(x_{i+1})$ para alguna función f resumen resistente a las preimágenes. Para iniciar el sistema, solo es necesario almacenar x_{1000} y el índice de la contraseña que se utilizará en la siguiente conexión.

A continuación veremos con más detalle el funcionamiento del protocolo TESLA.

2.1.1. TESLA

La autenticación de mensajes en *multicast* o *broadcast* requiere una fuente de asimetría, de forma que los receptores puedan verificar la autenticidad de los mensajes que reciben pero no puedan utilizar esta información para enviar ellos mismos datos suplantando la fuente de información auténtica. TESLA utiliza el tiempo para crear esta asimetría. Todos los participantes en una comunicación deben estar ligeramente sincronizados (dentro de cierto margen de tolerancia, todas las partes se tienen que poner de acuerdo en el tiempo actual). Entonces, el protocolo TESLA opera de la manera siguiente:

1) El emisor divide un segmento de tiempo en intervalos uniformes.

Cadena resumen

La cadena resumen se asigna a un segmento de tiempo en el orden inverso a su generación, de manera que cualquier valor de un intervalo se puede utilizar para obtener los valores de los intervalos de tiempo anteriores pero no proporciona ninguna información de los valores de los intervalos posteriores.

2) El emisor construye una cadena resumen y asigna los valores de la cadena $v_0,v_1,\ldots,v_N$ a los intervalos de tiempo de forma secuencial (un valor para cada intervalo de tiempo). Durante la inicialización del sistema, el emisor envía el valor v_0 a través de un sistema de autenticación fuerte tradicional, como puede ser una firma digital.

3) Cuando el emisor quiere enviar un paquete, calcula el MAC de sus contenidos utilizando como clave criptográfica el valor v_i asociado al intervalo de tiempo en el que está operando. Después envía el paquete de datos, el MAC, y una referencia al intervalo de tiempo en el que ha estado operando.

4) El emisor hace público el valor v_i utilizado en el intervalo t_i después de un margen predefinido de tiempo (por ejemplo, la clave utilizada en el intervalo de tiempo t_i se revela en el tiempo ti+2).

5) Cuando un receptor recibe un mensaje de datos, comprueba que la clave utilizada para calcular el MAC sigue siendo secreta (puede comprobarlo porque tiene el reloj sincronizado con el emisor y sabe en qué intervalos de tiempo

se publican las claves). Si la clave todavía es secreta, nada más recibirla comprueba que es auténtica, es decir, que realmente pertenece al emisor que dice. Puede hacerlo aplicando recursivamente una función resumen sobre la clave v_i recibida hasta encontrar el valor v_0 que ha sido autenticado por el usuario por medio de una firma digital, o hasta encontrar otro valor v_j, con $j < i$, que ya haya sido verificado antes por el receptor.

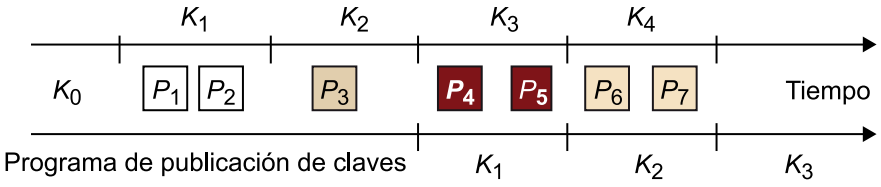
Una vez que se publica la clave v_i, un adversario que monitorizase la comunicación podría utilizarla para fabricar mensajes falsos. Sin embargo, estos mensajes no serían aceptados por el sistema porque los usuarios saben que en el intervalo de tiempo actual t_j el uso de la clave v_i, $i < j$, ya ha caducado.

La figura 1 muestra un ejemplo de cómo se utilizan las claves de TESLA para firmar diferentes paquetes, y cómo se revelan transcurrido un tiempo determinado. Los paquetes P_1 y P_2 se autentican utilizando K_1. La clave K_1 no se revela hasta después de dos intervalos de tiempo, y cuando se revela, ya no es válida, sino que la que es válida es la clave K_3.

Nota

Hay una versión más ligera del protocolo TESLA, llamada μ-TESLA, adecuada para redes de sensores.

Figura 1. Autenticación TESLA



El protocolo TESLA es apropiado para redes de comunicación inalámbricas que tienen una fiabilidad menor que las redes cableadas. Observad que si durante algún intervalo de tiempo se pierden las claves y no se publican, el receptor podrá recuperarlas a partir de algún valor que reciba posteriormente, y a partir de aquí podrá verificar la autenticidad de todos los paquetes recibidos.

2.2. Encaminamiento

Una de las características particulares de las redes *ad hoc* es su naturaleza multisalto. Los protocolos de encaminamiento multisalto se pueden clasificar en tres categorías:

- los que se basan en la topología de la red,
- los que se basan en la posición de los nodos, y
- los que tienen una aproximación híbrida entre las dos anteriores.

Los primeros tienen los mismos principios que los protocolos tradicionales de Internet (crear y mantener tablas de encaminamiento según la distribución de los nodos, distribuir la información de enlace, etc.), mientras que los segundos utilizan la localización física de los nodos para encaminar paquetes a su destino. La ventaja de los protocolos basados en la posición es que no necesitan mantener información de encaminamiento o descubrir rutas explícitamente.

Los nodos conocen su localización gracias a un servicio de posicionamiento (por ejemplo, GPS) y obtienen la posición de los otros nodos gracias a un servicio de localización. Cuando quieren enviar un paquete, la fuente obtiene la localización del destino e incluye esta información en la cabecera del paquete. Después los nodos intermedios toman decisiones de encaminamiento basándose en su posición y la localización del destino. Del mismo modo, los protocolos basados en la posición plantean el problema de que los nodos tienen que disponer de un hardware caro, como el GPS, y además se vulnera totalmente la privacidad de los usuarios en una comunicación al revelarse dónde están posicionados. Por este motivo, los protocolos basados en la posición están relegados a aplicaciones muy específicas. La gran mayoría de los servicios se fundamentan en protocolos basados en la topología, y estos son los que analizaremos en este apartado.

El grupo de trabajo en redes *ad hoc* de la Internet Engineering Task Force (IETF), llamado grupo MANET, está desarrollando varios protocolos de encaminamiento basados en la topología. Estos protocolos están fundamentados en dos enfoques diferentes:

- **Protocolos reactivos**. Estos protocolos establecen las rutas necesarias bajo demanda, es decir, la información sobre la topología de la red solo se envía cuando hace falta. Cuando un nodo quiere comunicarse con otro con el que no hay una ruta establecida, se activa el protocolo de establecimiento para generar una nueva ruta. Los ejemplos más conocidos de este tipo de protocolos son los denominados *dynamic source routing protocol* (DSR) y *ad hoc on-demand distance vector* (AODV).

- **Protocolos proactivos**. Después de un paso inicial de establecimiento, estos protocolos mantienen las rutas activas y "limpias" (sin bucles) constantemente. Cuando un nodo quiere comunicarse con otro, ya tiene lista la ruta de enlace por donde enviar paquetes. El ejemplo más conocido de este modelo de protocolos es el *optimized link state routing protocol* (OLSR).

A continuación veremos de forma más detallada estos protocolos y estudiaremos las vulnerabilidades de seguridad que presentan y cómo se pueden solucionar.

2.2.1. DSR

El DSR es uno de los primeros protocolos propuestos para el encaminamiento de redes *ad hoc* y uno de los más influyentes sobre los protocolos que han ido surgiendo desde entonces.

Lectura complementaria

El protocolo DSR está definido en una RFC:

D. Johnson; Y. Hu; D. Maltz (2007). "The dynamic source routing protocol (dsr) for mobile ad hoc networks for IPv4". *RFC 4728* (*experimental*) (febrero).

Lo que define el DSR es que se trata de un protocolo pensado para que la fuente de datos establezca el encaminamiento de los paquetes. Así, cada paquete lleva en la cabecera una lista de los nodos por los que tiene que pasar para llegar al destino deseado. Cuando un nodo recibe un paquete, primero verifica si él es su destinatario y, de no ser así, comprueba si está en la lista de transportadores del paquete. Si se trata de un nodo encaminador, retransmite el paquete al siguiente nodo de la lista (que tiene que ser un vecino directo), o de lo contrario descarta el paquete.

Cuando se utiliza encaminamiento de fuente, el nodo origen de la transmisión tiene que conocer la ruta completa hasta el destino antes de enviar cualquier paquete. En general, los protocolos de encaminamiento están formados básicamente por dos mecanismos:

- **Descubrimiento de rutas**. Se utiliza cuando los nodos no tienen información de cómo hacer que un paquete llegue a su destino. El mecanismo de descubrimiento de rutas en DSR se basa en la inundación de toda la red con un mensaje de petición de ruta[9] y en el retorno de las respuestas de ruta[10].

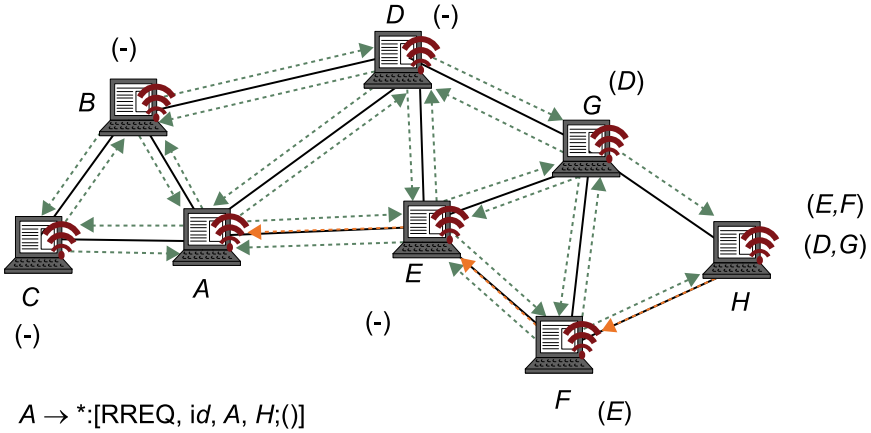
- **Mantenimiento de rutas**. En este caso, los nodos de la red ya disponen de tablas de encaminamiento que contienen cierta información de la topología de la red y de cómo pueden encaminar la información. Este mecanismo permite detectar errores de ruta (por ejemplo, si un enlace de la ruta ya no funciona porque un nodo se ha movido o ha salido de la red).

[9] En inglés, *route request* (RREQ).

[10] En inglés, *route reply* (RREP).

La figura 2 ilustra el funcionamiento del descubrimiento de rutas en DSR para un nodo origen *A* y un nodo destino *H*. El nodo origen envía una RREQ con un identificador de petición (*id*), el identificador del nodo origen (*A*), el del destino (*H*), y una lista vacía de nodos intermedios. Lo envía en *broadcast* a todos los nodos que están bajo su cobertura. Cada nodo que recibe una copia de la petición verifica que no la ha recibido ya antes. Si la petición ya ha sido recibida, la descarta. En caso contrario, el nodo añade su identificador a la lista de nodos por los que ha pasado la RREQ y reenvía el paquete en *broadcast* a sus vecinos. Este procedimiento se repite hasta que la petición llega al destino *H*.

Figura 2. Descubrimiento de rutas DSR



$A \rightarrow$ *:[RREQ, id, A, H;()]
$B \rightarrow$ *:[RREQ, id, A, H;(B)]
$C \rightarrow$ *:[RREQ, id, A, H;(C)]
$D \rightarrow$ *:[RREQ, id, A, H;(D)]
$E \rightarrow$ *:[RREQ, id, A, H;(E)]
$F \rightarrow$ *:[RREQ, id, A, H;(E,F)]
$G \rightarrow$ *:[RREQ, id, A, H;(D,G)]

$H \rightarrow A$:[RREP, (camino F,E); (E,F)]

Cuando *H* recibe el paquete, extrae la ruta que lo une con el nodo *A* de la lista de identificadores de la RREQ, la invierte y la copia en su tabla de encaminamiento. Entonces genera una RREP copiando la lista de identificadores de la RREQ en la respuesta. En ese momento, la respuesta es enviada por unidestino (*unicast*) a la fuente a partir de la ruta que el nodo ha guardado en su tabla de encaminamiento.

Durante la fase del mantenimiento de rutas, cada nodo intermedio de una comunicación se tiene que asegurar de que el paquete que está enviando realmente llega al salto siguiente. Esto se consigue requiriendo que el protocolo de la capa de enlace envíe un ACK para cada paquete que se entregue a un nodo. También es posible implementar que los nodos escuchen las transmisiones de sus vecinos y se aseguren de que el paquete que cada uno envía es reenviado a su vez por el nodo vecino.

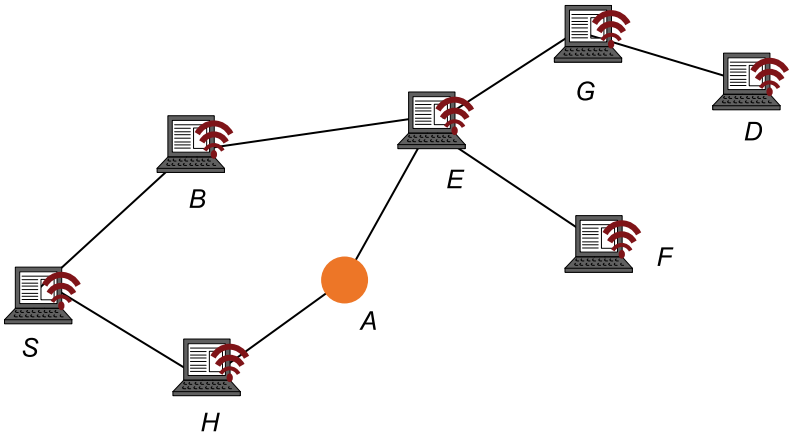
Vulnerabilidades del DSR

A continuación exponemos algunas de las vulnerabilidades más importantes del DSR. Utilizaremos la figura 3 para ilustrar algunos ataques contra el protocolo de encaminamiento:

1) Disrupción de rutas. Supongamos que un nodo *S* quiere encontrar una ruta hacia el destino *D*. *S* inicia el protocolo de descubrimiento de rutas inundando la red con una RREQ. Supongamos que los nodos tienen la memoria caché vacía. El nodo malicioso *A* puede prevenir el descubrimiento de la ruta *S*, *H*, *A*, *E*, *G*, *D* eliminando el mensaje de RREQ que proviene de *S* o eliminando el mensaje de respuesta RREP. Por otro lado, el nodo *A* también puede hacer que el nodo *E* reciba el mensaje de RREQ a partir de *A* antes que a partir de *B*. El

nodo *A* puede, por ejemplo, mantener el canal constantemente ocupado para evitar que *E* reciba nada de *B*. De este modo, el adversario puede impedir que se descubra cualquiera de las rutas existentes entre *S* y *D*.

Figura 3. Topología de una red donde hay un nodo malicioso A



2) Desvío de rutas (ataque *sinkhole*). De nuevo asumimos que *S* quiere encontrar una ruta hacia *D* e inicia el protocolo DSR enviando una RREQ. Cuando *A* recibe la petición de *H*, *A* responde con una falsa RREP que contiene la ruta *S*, *H*, *A*, *D*. La ruta falsa es enviada hacia *S*. Dado que la ruta falsa es más corta que la descubierta a través del nodo *E*, *S* decide usar la ruta *S*, *H*, *A*, *D*. De este modo, el adversario consigue modificar la ruta natural entre *A* y *D* y escuchar todos los mensajes que se transmiten.

3) Creación de estados de encaminamiento incorrectos. En el caso del DSR, crear estados de encaminamiento incorrectos significa que el adversario engaña a la fuente de una RREQ haciéndole aceptar y poner en la memoria caché una ruta inexistente hacia el destino. Por ejemplo, cuando *S* inicia un descubrimiento de rutas, *A* puede no reenviar el mensaje recibido por *H*. Simplemente, espera hasta que le llega otra copia de la misma petición a través del nodo *E*. Esta copia contiene la ruta *S*, *B*, *E*. Después, A genera un mensaje RREP que contiene la ruta *S*, *B*, *E*, *A*, *D* y lo envía a *E*. La respuesta es enviada de vuelta hacia *S* a través del nodo *B*, y *S* guarda la ruta *S*, *B*, *E*, *A*, *D*.

Existen varios protocolos que añaden seguridad al DSR, como por ejemplo los siguientes:

- SRP (*on-demand source routing*),
- Ariadne (*on-demand source routing*),
- endairA (*on-demand source routing*),
- SADSR (*security aware adaptive DSR*), y
- SRDP (*secure route discovery protocol*).

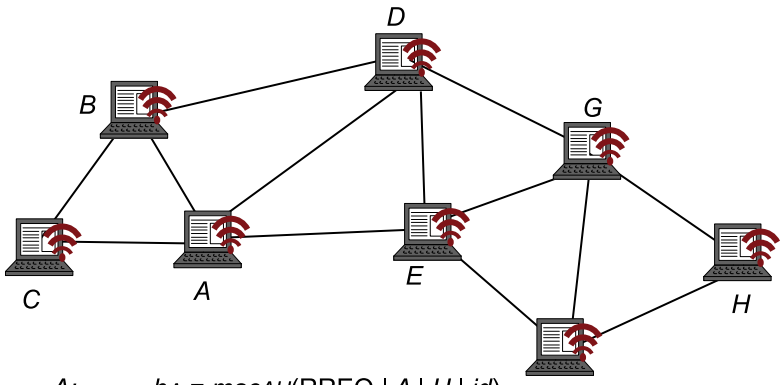
A continuación estudiaremos el funcionamiento de dos de los más relevantes, Ariadne y endairA.

Ariadne

Ariadne utiliza la autenticación de los mensajes de control RREQ y RREP para evitar las modificaciones y falsificaciones. La autenticación de los mensajes se puede llevar a cabo de tres formas diferentes: a partir de firmas digitales, usando MAC o a partir del protocolo TESLA. Además, Ariadne utiliza un mecanismo basado en funciones resumen para evitar la manipulación de la información por parte de los nodos intermedios de una comunicación durante el proceso de envío de las RREQ.

La operación de Ariadne utilizando firmas digitales se ilustra en la figura 4. En esta figura se muestra un protocolo de descubrimiento de rutas entre los nodos *A* y *H* (el mismo que se ilustra en la figura 2). La fuente *A* genera una RREQ y la envía por *broadcast* a toda la red. Este mensaje de RREQ, además de contener la información básica del DSR, contiene un código de autenticación MAC calculado con una clave compartida entre la fuente *A* y el destino *H*. El paquete RREQ va viajando a través de los diferentes nodos de la red. Cada vez que un nodo retransmite un paquete RREQ, calcula de forma iterativa un resumen *hash* sobre el último MAC o resumen *hash* que ha recibido junto con su identificador de nodo. Además, el nodo que recibe una RREQ añade su identificador a la lista de nodos itermedios y adjunta una firma de todos los valores del mensaje.

Figura 4. Protocolo Ariadne con operaciones de firma digital



A: $h_A = mac_{AH}(\text{RREQ} \mid A \mid H \mid id)$
$A \rightarrow *$: [RREQ, A, H, id, h_A, (), ()]

E: $h_E = H(E \mid h_A)$
$E \rightarrow *$: [RREQ, A, H, id, h_E, (E), (sig_E)]

F: $h_F = H(F \mid h_E)$
$F \rightarrow *$: [RREQ, A, H, id, h_F, (E,F), (sig_E, sig_F)]

$H \rightarrow A$: [RREP, H, A, (E,F), (sig_E, sig_F), sig_H]

Cuando el destino recibe la RREQ, verifica el valor de resumen recalculando el MAC de la fuente y los valores de resumen salto a salto. Después verifica todas las firmas digitales. Si todas estas verificaciones son exitosas, el destino genera una RREP y la envía de vuelta a la fuente a través de la ruta inversa obtenida en la RREQ. La RREP contiene los identificadores del nodo destino

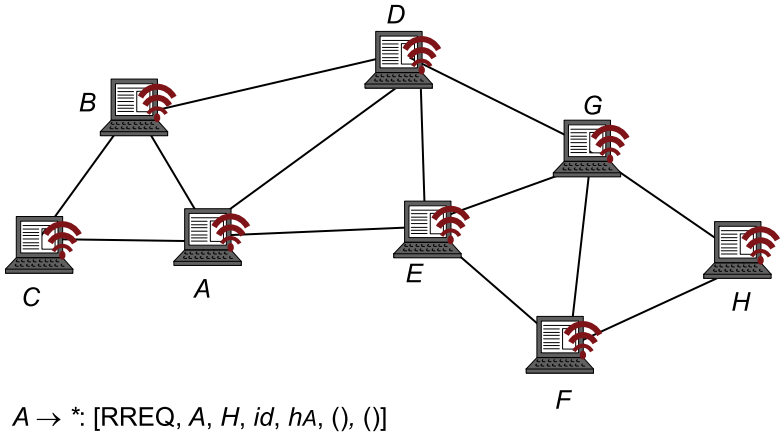
y del nodo fuente, la ruta que hay que seguir, la lista de firmas obtenidas de la RREQ y la firma digital del destino sobre todos estos elementos. Cada nodo intermedio pasa la respuesta al siguiente nodo de la ruta sin efectuar modificaciones. Cuando la fuente recibe la respuesta, verifica la firma digital de los nodos intermedios (para ello necesita reconstruir las peticiones que firmaron los nodos intermedios). Si las verificaciones son correctas, la fuente recibe la respuesta como válida.

Cuando Ariadne utiliza MAC, se asume que cada nodo intermedio comparte una clave con el nodo destino. El funcionamiento es muy similar a Ariadne, con firmas digitales, pero en lugar de firmas, los nodos intermedios calculan MAC de la RREQ con la clave que comparten con el destino (en el caso del ejemplo, el nodo *H*). Cuando la RREQ llega al destino, puede verificar todos los MAC y calcular otro MAC de todos los campos con la clave que comparten destino y origen. Observa que, en este caso, la fuente no puede autenticar los nodos intermedios, sino que tiene que confiar en que el destino ha ejecutado su parte del protocolo correctamente. Por otro lado, los nodos intermedios no pueden autenticar ni la fuente ni el destino.

Si se utiliza Ariadne con TESLA, los usuarios van añadiendo MAC al paquete de forma similar a las operaciones de Ariadne con MAC. La ventaja es que las claves MAC permiten autenticar los paquetes y son mucho más eficientes que las firmas digitales. El punto negativo es el retardo que introduce TESLA, que, según el tipo de servicios, puede ser insostenible.

La figura 5 muestra un ejemplo del protocolo Ariadne con modo de operación TESLA. Observad que en la RREP, los nodos intermedios retardan la respuesta hasta que pueden revelar la clave TESLA que han utilizado para generar el MAC. Entonces incluyen la clave TESLA en la respuesta. La fuente puede verificar la firma MAC del destino y de todos los nodos intermedios.

Figura 5. Protocolo Ariadne con modo de operación TESLA



$A \rightarrow *$: [RREQ, A, H, id, h_A, (), ()]
$E \rightarrow *$: [RREQ, A, H, id, h_E, (E), ($mac_{KL,i}$)]
$F \rightarrow *$: [RREQ, A, H, id, h_F, (E,F), ($mac_{KE,i}$, $mac_{KF,i}$)]

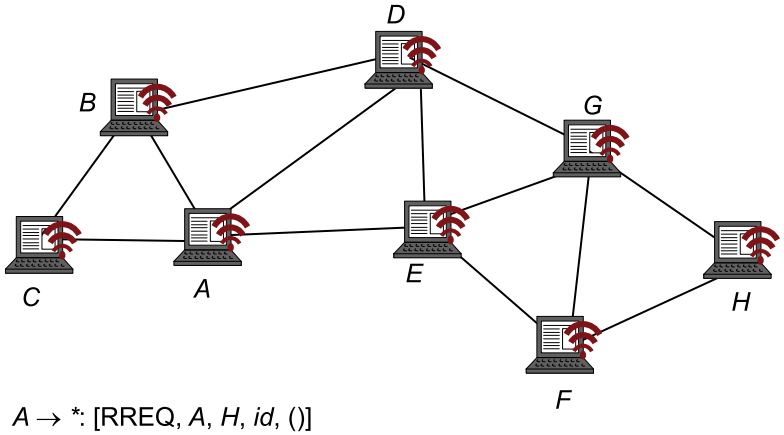
$H \rightarrow F$: [RREP, H, A, (E,F), ($mac_{KE,i}$, $mac_{KF,i}$), mac_{HA}, ()]
$F \rightarrow E$: [RREP, H, A, (E,F), ($mac_{KF,i}$, $mac_{KF,i}$), mac_{HA}, ($K_{F,i}$)]
$E \rightarrow A$: [RREP, H, A, (E,F), ($mac_{KE,i}$, $mac_{KF,i}$), mac_{KHA}, ($K_{F,i}$,$K_{F,i}$)]

endairA

endairA es otra extensión de seguridad para el protocolo DSR que está inspirada en el protocolo Ariadne. La principal diferencia con su predecesor es que, en lugar de firmar los mensajes de RREQ, en endairA los nodos intermedios firman la RREP. Esto explica el nombre endairA, palíndromo de Ariadne.

El funcionamiento de endairA se ilustra en la figura 6. La fuente genera una RREQ que contiene los identificadores de la fuente y del destino, así como un identificador de RREQ aleatorio. Cada nodo intermedio que recibe una RREQ por primera vez añade su identificador a la ruta acumulada hasta aquel momento y retransmite el mensaje en difusión (*broadcast*). Cuando la petición llega al destino, se genera una RREP. La RREP contiene los identificadores de la fuente y del destino, la ruta acumulada desde el nodo origen y la firma digital del destino en todos estos elementos. La respuesta se envía de vuelta al nodo origen a través del camino de nodos encontrado en la RREQ. Cada nodo intermedio que recibe la respuesta verifica que su identificador está en la lista y que el identificador del nodo anterior y posterior pertenecen a nodos vecinos. También verifica que las firmas digitales de la respuesta son válidas y que corresponden a los identificadores de los nodos de la lista y al nodo destino. Si estas verificaciones fallan, entonces la respuesta es descartada. En caso contrario, se firma y se pasa el paquete al siguiente nodo de la ruta hacia la fuente. Cuando la fuente recibe la RREP, verifica que el mensaje procede de un vecino, comprueba las firmas y, si todo es correcto, acepta la ruta que se indica en el mensaje.

Figura 6. Protocolo endairA



$A \rightarrow *$: [RREQ, A, H, id, ()]
$E \rightarrow *$: [RREQ, A, H, id, (E)]
$F \rightarrow *$: [RREQ, A, H, id, (E,F)]

$H \rightarrow F$: [RREP, H, A, (E,F), (sig_H)]
$F \rightarrow E$: [RREP, H, A, (E,F), (sig_H, sig_F)]
$E \rightarrow A$: [RREP, H, A, (E,F), (sig_H, sig_F, sig_E)]

Una de las principales contribuciones de endairA es su gestión eficiente de los cálculos criptográficos pesados. Todo el proceso criptográfico se hace en los mensajes de respuesta, lo que significa que solo se ven involucrados en este proceso los nodos que forman parte de la ruta real entre la fuente y el destino.

Cálculo criptográfico en Ariadne

En el caso de Ariadne, todos los nodos de la red tenían que hacer el cálculo criptográfico, ya que el descubrimiento de rutas se hace con una inundación de paquetes por toda la red.

Un problema básico de endairA es que resulta vulnerable a ataques de inundación por petición de rutas. Puesto que los mensajes RREQ no están autenticados, cualquier adversario puede iniciar un proceso de descubrimiento de rutas. Esto se puede solucionar indicando que las RREQ tienen que ir firmadas por el nodo origen, con la subsiguiente carga para el sistema.

2.2.2. AODV

El protocolo AODV[11] es un protocolo reactivo basado en las tradicionales tablas de encaminamiento, no en el encaminamiento de fuente como DSR. Fue definido en el 2003 por la IETF. En el 2005, la IETF comenzó a trabajar en una nueva propuesta de protocolo reactivo y basado en tablas de encaminamiento, DYMO[12]. DYMO es una clara evolución del protocolo AODV, por eso las últimas propuestas de este protocolo se han denominado AODVv2. Actualmente, AODVv2 está en fase de borrador, en previsión de convertirse en el estándar de encaminamiento bajo demanda de la IETF.

(11)Del inglés, *ad-hoc on-demand distance vector*.

(12)Del inglés, *dynamic MANET on-demand routing protocol*.

Lectura complementaria

El protocolo AODV está definido en una RFC:

C. Perkins; E. Royer; S. Das (2003). "Ad hoc on-demand distance vector (AODV) routing". *RFC 3561* (*experimental*).

AODV opera de forma similar a DSR, con un mecanismo para hacer el descubrimiento de rutas y otro para el mantenimiento. La diferencia estriba en que los nodos utilizan tablas de encaminamiento. Una entrada de una tabla de encaminamiento contiene la siguiente información:

- identificador del destino,
- número de saltos requeridos para llegar al destino,
- identificador del nodo del salto siguiente de la ruta hacia el destino, y
- número de secuencia del destino.

Número de secuencia del destino

Los números de secuencia del destino sirven para identificar y descartar información no actual y asegurar que el protocolo no entra en bucles de encaminamiento.

En AODV, cuando un nodo quiere enviar información a un destino y no tiene una entrada válida para este nodo en su tabla de encaminamiento, lo que hace es generar una petición de ruta RREQ, que contiene los identificadores de la fuente y el destino, un contador del número de saltos (inicialmente a cero), un número de secuencia asociado a la fuente y un número de secuencia asociado al destino. Cada nodo tiene un único número de secuencia, que se incrementa después de que se detecte un cambio en el conjunto de vecinos del nodo. La RREQ también contiene un número que identifica la petición con el mismo objetivo que en el protocolo DSR, ayudar a los nodos intermedios a detectar duplicados de la misma petición y descartarlos. La RREQ se envía en difusión a todos los vecinos.

Cuando un nodo intermedio recibe una RREQ, primero identifica si se trata de un duplicado o no. Si es un duplicado, lo descarta. En caso contrario, el nodo comprueba si tiene una entrada válida en su tabla de rutas para el destino indicado en la petición. Si no la tiene, o si tiene una entrada válida pero con un número de secuencia de destino menor al que consta en la petición, incrementa el contador de saltos y reenvía el mensaje a sus vecinos. Por otro lado, si el nodo intermedio tiene una entrada válida, envía una respuesta RREP. Si la petición llega al destino, evidentemente, esta también genera una respuesta RREP.

Además de las operaciones descritas en el párrafo anterior, cuando un nodo recibe una RREQ no duplicada, también crea o actualiza la entrada de su tabla de encaminamiento que corresponde al nodo fuente de la petición. Si la entrada de la tabla ya está creada y el número de secuencia de la fuente es mayor que el que aparece en la RREQ, o si el número es igual pero la longitud de la nueva ruta a la fuente es más larga que la que ya tiene el nodo (esto se puede comprobar a partir del valor del número de saltos que aparece en la RREQ), la tabla de encaminamiento del nodo no se actualiza. Si la entrada de la tabla sí que se crea o se actualiza, la información que hay que introducir es la siguiente:

- el identificador de la entrada, que corresponde al identificador de la fuente de la RREQ;

- la longitud de la ruta, que se establece con el valor del contador de saltos de la RREQ;

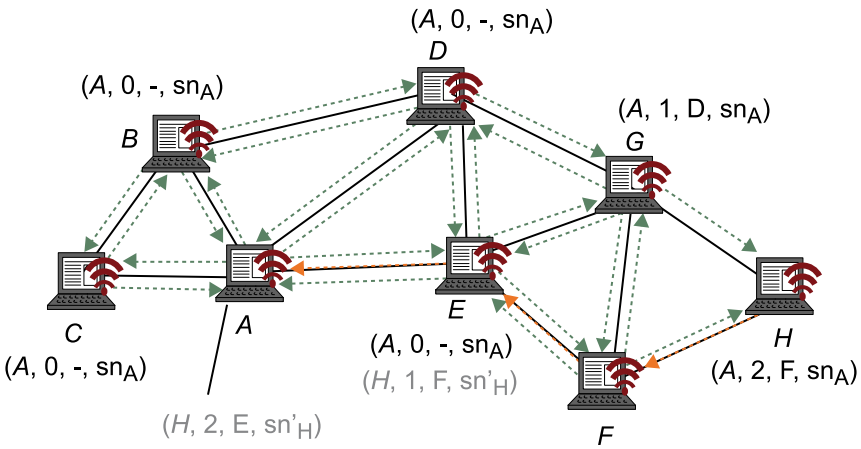
- el salto siguiente para llegar al nodo, que es el identificador del nodo a través del cual se ha recibido el mensaje de RREQ, y

- el número de secuencia de la entrada, que es el número de secuencia de la fuente de la petición. Esta entrada de la tabla de encaminamiento solo se utilizará si este nodo recibe finalmente un mensaje de respuesta RREP para ser reenviado a la fuente.

Como se ha mencionado, tanto el nodo destino como un nodo intermedio que tenga información actualizada del destino en su tabla de ruta pueden construir la RREP. La RREP contiene información sobre el número de secuencia del destino y la distancia (en saltos) a que se encuentra. El mensaje de RREP se reenvía hacia el nodo origen a través del camino que se ha establecido por medio de la RREQ. El procesado que llevan a cabo los nodos intermedios del mensaje RREP es muy similar al que efectúan para el mensaje RREQ; es decir, incrementan el contador de saltos y crean/actualizan la entrada de su tabla de encaminamiento que corresponde al nodo destino.

Finalmente, AODV también dispone de mecanismos de mantenimiento de rutas que usan mensajes de error similares a los de DSR si encuentran que algún enlace de la red está roto.

Figura 7. Protocolo AODV entre los nodos *A* y *F*



$A \rightarrow$ *:[RREQ, id, A, H, 0, sn_A, sn_H]
$B \rightarrow$ *:[RREQ, id, A, H, 1, sn_A, sn_H]
$C \rightarrow$ *:[RREQ, id, A, H, 1, sn_A, sn_H]
$D \rightarrow$ *:[RREQ, id, A, H, 1, sn_A, sn_H]
$E \rightarrow$ *:[RREQ, id, A, H, 1, sn_A, sn_H]
$F \rightarrow$ *:[RREQ, id, A, H, 2, sn_A, sn_H]
$G \rightarrow$ *:[RREQ, id, A, H, 2, sn_A, sn_H]

$H \rightarrow F$:[RREP, A, H, 0, sn'_H]
$F \rightarrow E$:[RREP, A, H, 1, sn'_H]
$E \rightarrow A$:[RREP, A, H, 2, sn'_H]

Vulnerabilidades del AODV

A continuación exponemos algunas de las mayores vulnerabilidades de AODV. Como en el caso de DSR, utilizaremos la figura 3 para ilustrar algunos ataques contra el protocolo de encaminamiento.

1) Interrupción de rutas. Supongamos que un nodo *S* quiere encontrar una ruta hacia el destino *D*. Un adversario puede evitar el descubrimiento de la ruta entre estos dos nodos manipulando el valor del número de saltos en el mensaje de RREQ. Por ejemplo, si el nodo *A* define que el número de saltos de la petición de ruta recibida del nodo *H* es cero, el nodo *E* pensará que la ruta más corta hacia *S* es a través de *A* y, por tanto, *E* enviará los mensajes de respuesta RREP destinados a *S* a través de *A*. Si *A* elimina los mensajes de RREP, la ruta entre *S* y *D* no se podrá descubrir nunca.

2) Desvío de rutas. El desvío de rutas en AODV también se puede llevar a cabo modificando el valor del número de saltos de los mensajes RREQ y RREP. Por ejemplo, si el nodo *A* de la figura no elimina el mensaje RREP en el anterior ataque de interrupción de rutas, entonces la ruta *S*, *H*, *A*, *E*, *G*, *D* se establecerá como oficial (cuando en realidad hay una más corta).

3) Creación de estados de encaminamiento incorrectos. Un nodo puede crear tablas de encaminamiento incorrectas si manipula el número de secuencia del destino o el contador de saltos en los mensajes de control del encaminamiento. Por ejemplo, si *S* inicia una RREQ hacia *D* y *A* recibe esta petición a través de *H*, *A* puede incrementar el número de secuencia del destino antes de reenviar el mensaje por difusión. Como resultado, *E* fija a *F* como su siguiente salto hacia *S*, ya que *F* es el vecino de *E*, y la ruta falsificada parece más fresca que la correcta que provenía del nodo *B*. Por tanto, *E* obtiene un estado incorrecto, ya que no hay ninguna ruta a *S* vía *F*.

Existen varios protocolos que añaden seguridad a AODV, como los siguientes:

- SAODV (*secure AODV*),
- SEAODV (*security enhanced AODV*),
- ARAN (*authenticated routing for ad hoc networks*), y
- SEDYMO (*secure DYMO*).

A continuación veremos el funcionament de SAODV, una de las propuestas más eficientes de las presentadas. SAODV es una extensión del protocolo básico AODV. Como hemos comentado, a partir de AODV han ido surgiendo mejoras denominadas DYMO y AODV2. La extensión que permitiría dar seguridad a estos protocolos mejorados sería SEDYMO, que se basa en los mismos principios de fucionamiento que SAODV, pero que además tiene en cuenta las funcionalidades extendidas de DYMO.

SAODV

Podemos considerar que los mensajes de encaminamiento de AODV tienen dos partes, una mutable (la información cambia a cada salto de la RREQ/RREP) y una inmutable. La parte mutable de los mensajes es la métrica, es decir, la información sobre la distancia (número de saltos) a la que se encuentran el

nodo origen o destino. La parte inmutable incluye el número de secuencia de los nodos, las direcciones de la fuente y el destino y el identificador de la petición.

La seguridad en SAODV se basa en dos mecanismos: las cadenas resumen para los campos mutables del mensaje y las firmas digitales para proteger la parte inmutable y creada por la fuente de la RREQ/RREP.

El mecanismo de cadenas resumen en SAODV hace que se deban añadir tres campos específicos para este uso en los mensajes de encaminamiento, de manera que la información final sobre la métrica es la siguiente:

- `HopCount`: contador del número de saltos. Inicialmente está a cero.

- `MaxHopCount`: valor que determina el número máximo de saltos que un paquete puede hacer en la red. Generalmente, se inicializa con el valor del TTL *(time to live)*.

- `Hash`: valor de la cadena resumen. Este campo se inicializa con un valor aleatorio llamado semilla *s*.

- `TopHash`: valor resultante de aplicar `MaxHopCount` veces la función resumen a la semilla *s*.

Los campos `MaxHopCount` y `TopHash` pertenecen a la parte inmutable del mensaje (y están protegidas por la firma digital), mientras que los campos `HopCount` y `Hash` cambian con cada salto. Cada nodo que recibe un mensaje de encaminamiento calcula la diferencia d entre los valores de los campos `MaxHopCount` y `HopCount`, y a continuación calcula *d* veces una función resumen sobre el campo `Hash`. El mensaje se valida de forma satisfactoria si el resultado de la operación coincide con el valor del campo `TopHash`. En ese caso, el nodo incrementa en una unidad el valor del `HopCount`, actualiza el valor del `Hash` calculando el resumen (*hash*) del anterior valor que había en ese campo y reenvía el paquete en difusión a sus vecinos.

El propósito de este mecanismo es que todo el mundo pueda verificar el valor del `HopCount` y que se puedan evitar ataques derivados de la falsificación de dicho valor. Sin embargo, este sistema solo permite evitar que un adversario reduzca de forma desmesurada el valor del número de saltos de un paquete (con lo que las rutas parecerían más cortas de lo que son en realidad), pero no que lo mantenga igual o lo aumente. Si un adversario no incrementa el valor del `HopCount`, en el fondo también está haciendo que esa ruta sea una unidad menor de lo que debería ser, con las consecuencias que este hecho puede tener para el encaminamiento. Por otro lado, si los usuarios aumentan el valor del

`HopCount`, pueden crear ataques de egoísmo (*selfishness*), lo que evitaría que las rutas de la red *ad hoc* pasasen por ellos para no sufrir el coste que comporta participar en una red colaborativa.

Comos hemos comentado, el otro mecanismo de seguridad de SAODV es la firma digital. El nodo emisor de la RREQ genera una firma utilizando su clave privada, que protege la integridad y la autenticidad de los parámetros estáticos del mensaje, y de este modo prevé ataques de modificación de los paquetes.

2.2.3. OLSR

El protocolo OLSR[13] es un protocolo proactivo que se diferencia de DSR y AODV en tanto que mantiene la información sobre las rutas activas constantemente actualizada. Esto evita que haya retardos al inicio de las transmisiones cuando un nodo quiere establecer una comunicación con un destino remoto. En contrapartida, requiere un flujo constante de pequeños mensajes de control que mantengan actualizada la información sobre el estado y la localización de los nodos de la red.

(13)Del inglés, *optimized link state routing*.

Lectura complementaria

El protocolo OLSR está definido en una RFC:

T. Clausen; P. Jacquet (2003). "Optimized link state routing protocol (OLSR)". *RFC 3626* (*experimental*) (octubre).

OLSR inunda periódicamente la red con mensajes de control. Así, esta inundación se hace por medio de nodos seleccionados (los llamados *multipoint relays*, MPR) para optimizar al máximo la sobrecarga del sistema y reducir el número de retransmisiones de un mensaje de control. El conjunto de MPR de un nodo dado es el subconjunto de sus vecinos que han sido seleccionados para que puedan cubrir (en términos de cobertura radio) a todos los vecinos que están estrictamente a dos saltos de distancia del nodo.

En OLSR hay, básicamente, dos tipos de mensajes:

1) Mensajes *hello*: como hemos visto, se trata de mensajes que se envían en difusión local para descubrir vecinos.

2) Mensajes TC (control de topología): se trata de mensajes que inundan la red a través de los MPR.

Los mensajes *hello* enviados por un nodo A determinado contienen una lista de sus supuestos vecinos. Para cada vecino de la lista, se detalla el estado de enlace entre el nodo A y él y si se trata de un nodo MPR o no.

Cuando un nodo *B* recibe un mensaje *hello*, obtiene diferente información: por un lado, entiende que el remitente del mensaje es un nodo vecino suyo. Si *B* está listado como vecino en el mensaje *hello*, entonces *B* aprende que *A* lo considera vecino, lo que significa que entre ellos hay un enlace bidireccional. De lo contrario, se asume que el enlace es asimétrico. Por otro lado, si *B* está

marcado como un MPR, entonces sabe que *A* lo ha seleccionado como tal y, por tanto, *A* se encuentra en el conjunto de selectores de *B*. Finalmente, mirando la lista de vecinos de *A*, *B* conoce los nodos que tiene a dos saltos de distancia.

Así pues, los mensajes *hello* en OLSR sirven para obtener información sobre el estado de los enlaces, para detectar vecinos a un salto y a dos saltos y para asignar los MPR. Indirectamente, los mensajes *hello* también sirven para hacer la selección de MPR, ya que los nodos se basan en la información de vecindad para determinar cuáles serán escogidos con este papel.

Solo los nodos MPR pueden enviar y retransmitir los mensajes TC. Contienen una lista de enlaces activos (como mínimo, entre el nodo remitente del MPR y sus selectores) y sirven para proporcionar suficiente información para que los nodos puedan construir su propia tabla topológica, y deducir entonces la tabla de rutas. Las rutas se calculan optimizando el número de saltos (por ejemplo, con el algoritmo del camino más corto de Dijkstra).

La información almacenada en una tabla de encaminamiento se basa en la vecindad y la topología, por lo que hay que recalcular si cambia alguna de estas informaciones. En particular, se debe actualizar en los casos siguientes:

- Si se detecta un cambio en la vecindad del nodo.
- Si expira una ruta y hay que actualizar esta información.
- Si se detecta un camino mejor (más corto) al mismo nodo de destino.

Vulnerabilidades de OLSR

OLSR es vulnerable a varios ataques. A continuación exponemos algunos de ellos:

1) Ataque de suplantación de identidades (*spoofing*). Los atacantes pueden usar el identificador de otro nodo y mostrarse a sí mismos como si fuesen otro. Así, el nodo *A* podría enviar un mensaje a su vecino *C* haciéndose pasar por *B*. Entonces, *C* consideraría que él y *B* son vecinos, cuando en realidad no lo son.

2) Disrupción de rutas. Un atacante puede eliminar los mensajes de encaminamiento que recibe en lugar de retransmitirlos a los demás vecinos como marca el protocolo. De esta manera se reduce la cantidad de información de encaminamiento disponible para los otros nodos.

3) Atac per replicació. En este ataque se reenvía a la red un mensaje de control antiguo y todos los nodos actualizan las tablas con información errónea.

Existen varios protocolos que añaden seguridad a OLSR, como por ejemplo:

- *secure OLSR*,
- *security aware OLSR* (SL-OLSR), y

- *shared secret-based OLSR*.

El objetivo de los protocolos seguros sobre OLSR es proporcionar servicios como autenticación de los nodos en los mensajes *hello*, proporcionar integridad a los mensajes de control y detectar nodos egoístas.

La mayoría de las soluciones se basan en lo siguiente:

1) Introducir un sello de tiempo en los mensajes para evitar los ataques de replicación.

2) Utilizar mecanismos de autenticación en los mensajes *hello* y TC. En la medida de lo posible, hay que tratar de evitar los mecanismos de clave pública por su elevado coste computacional. El problema reside en cómo llevar a cabo una gestión eficiente de claves para poder utilizar algoritmos de clave simétrica. Una solución es utilizar el protocolo TESLA.

Ved también

El protocolo TESLA se estudia en el subapartado 2.1.1 de este módulo.

3) Correlacionar los datos de los diferentes mensajes *hello* para evitar incoherencias. Para evitar ataques en la red de nodos internos (y autenticados), hay que intentar validar los datos que aportan los diferentes nodos de la red.

2.3. Privacidad

La propiedad de la privacidad significa que un sujeto puede controlar cuándo, dónde y cómo se usa la información sobre sí mismo y quién lo hace. La privacidad es uno de los grandes problemas de las redes *ad hoc*, ya que, para que dos nodos remotos puedan comunicarse, deben declarar que están activos en la red y tienen que participar en los protocolos de encaminamiento, con la consiguiente publicación de cuál es su posición/localización en la red.

Los ataques a la privacidad pueden ser tanto activos como pasivos. En los ataques activos, los usuarios maliciosos participan en el protocolo de red que pretenden romper, ya sea por medio de ataques externos (como usuarios ajenos al sistema) o con ataques internos (como miembros lícitos de la red). Por otro lado, los ataques pasivos no perturban el funcionamiento normal de los protocolos de red, los atacantes escuchan de manera no autorizada los paquetes que se transmiten por la red y, gracias a un análisis de tráfico, extrapolan información como las rutas de transmisión, el contenido de los mensajes o la identidad, posición o movimiento de los nodos.

La privacidad es un concepto muy amplio, pero en el caso de las comunicaciones en red, lo que nos interesa es focalizarnos principalmente en tres de sus propiedades:

1) Anonimato de los sujetos: propiedad de no ser identificable entre un conjunto de sujetos.

2) Desvinculación de mensajes: propiedad de ocultar la relación existente entre una comunicación y las personas que la llevan a cabo.

3) Indetectabilidad: incapacidad de distinguir si un elemento existe o no. En el caso de los mensajes, la indetectabilidad significa que estos no son lo bastante discernibles de, por ejemplo, ruido blanco.

A continuación veremos algunos mecanismos que permiten evitar las vulnerabilidades de privacidad en las redes *ad hoc*.

2.3.1. Anonimato de los sujetos

En este subapartado veremos dos mecanismos básicos que permiten proporcionar anonimato a los sujetos de una comunicación: los seudónimos y las funciones unidireccionales con trampa.

Seudónimos

Una forma de ocultar la identidad de los sujetos que actúan en una comunicación es con el uso de seudónimos, es decir, utilizando una etiqueta privada que permite, de manera discrecional, distinguir a los participantes de una transacción. A partir de la información pública de la red, los nodos son incapaces de generar y/o vincular seudónimos para el resto de los miembros de la red.

Lectura recomendada

Los esquemas basados en seudónimos fueron introducidos en 1985 por Chaum:

David Chaum (1985). "Security without identification: transaction systems to make big brother obsolete". *Communication ACM* (núm. 28, vol. 10, pág. 1030-1044.)

Las dificultades de poner en funcionamiento un sistema de seudónimos son las siguientes:

- **Temporalidad**. Los seudónimos se tienen que renovar periódicamente porque su uso revela cierta información que se podría utilizar para identificar o localizar a un sujeto.

- **Generación y gestión de los seudónimos**. El vínculo entre un seudónimo y la identidad real del sujeto o enlace a la que está asociado es privado. No obstante, hay que proporcionar mecanismos para hacer llegar esa información a los usuarios autorizados, de modo que la comunicación entre entidades sea viable.

- **Autenticación**. La autenticidad de los participantes en una transacción debería estar garantizada aunque se utilizasen seudónimos.

Los sistemas de seudónimos más comunes son los que utilizan una tercera entidad de confianza (TTP), responsable de generar, renovar, revocar y autenticar los seudónimos.

A pesar de que la TTP renueva los seudónimos periódicamente, este hecho por sí solo puede no proteger lo suficiente la red ante un potencial usuario malicioso con capacidad de escuchar todas las comunicaciones. La razón es que un atacante con capacidades globales de escucha podría enlazar con una alta probabilidad los diferentes seudónimos de un único nodo basándose en la posición y velocidad de la información que se origina en cierto espacio de la red. Es decir, una vez que el atacante conoce la localización de un nodo, su perfil de movimiento y la frecuencia con la que suele enviar mensajes, es muy fácil seguirle la pista aunque cambie de seudónimo.

Una forma de minimizar el efecto de los atacantes con gran poder de escucha es la inclusión de zonas de mezcla (*mix zone*) en la red. Esto significa que los nodos cambian de seudónimo cuando se encuentran en unas áreas predefinidas, pequeñas y acotadas. Este modelo busca la convergencia en tiempo y en espacio de varios nodos cambiando de seudónimo. Al haber diferentes usuarios ejecutando la misma operación al mismo tiempo, un adversario ya no puede mapear directamente cuál es el nuevo seudónimo de cada nodo y, por tanto, se incrementa la privacidad.

Funciones unidireccionales con trampa

Las funciones unidireccionales con trampa[14] son funciones unidireccionales $f : X \to Y$ tales que es fácil obtener $f(x)$ para cualquier $x \in X$, y que permiten el cálculo eficiente de la inversa (encontrar $x \in X$ tal que $f(x) = y$) si y solo si se tiene cierta información adicional, la trampa. En caso contrario, el cálculo de la inversa es computacionalmente intratable.

(14)En inglés, *trapdoor functions*.

Las funciones unidireccionales con trampa se utilizan para la identificación anónima de los receptores de una comunicación. El emisor envía la información de identificación de la comunicación oculta en una función unidireccional, de modo que solo el receptor legítimo de la transmisión, que posee la información trampa, sea capaz de recuperarla.

La forma más simple de implementar una función unidireccional para proporcionar anonimato de recepción es mediante criptografía de clave pública. La identidad del receptor se envía cifrada con la clave pública del mismo receptor, de modo que solo él pueda abrir con éxito el paquete. Sin embargo, esta solución resulta muy costosa, ya que el descubrimiento de rutas en redes *ad hoc* se hace por medio de mecanismos de inundación *broadcast*, y si todos los nodos que reciben un paquete tienen que hacer una operación criptográfica para descubrir si son los receptores de un paquete, la carga total del sistema es insostenible.

Nota

Protocolos de encaminamiento como SDAR y AnonDSR emplean criptografía de clave pública para proteger la identidad de los dos nodos de una comunicación.

Otra alternativa consiste en utilizar criptografía de clave simétrica. En este caso, se asume que origen y destino comparten una clave que se ha distribuido a través de un canal seguro. El origen cifra la identidad del destino y un número aleatorio con la clave simétrica que comparten. El nodo que pueda abrir este sobre y comprobar que está su identificador es el receptor legítimo. Finalmente, en el mensaje de respuesta al origen, el destino envía el número aleatorio del sobre como prueba de que lo ha recibido.

Nota

Los protocolos ANODR y ASR utilizan criptografía simétrica para ocultar la identidad del destino de una comunicación.

Finalmente, otra solución es utilizar funciones con trampa más ligeras, basadas en funciones resumen. Si el origen y el destino comparten un secreto, el destino se puede identificar mediante un valor HMAC de cierto valor aleatorio (que puede ser público).

2.3.2. Desvinculación de mensajes

Las principales técnicas que se utilizan para evitar que un adversario pueda inferir los sujetos que participan en una comunicación se basan en un modelo de mezclador (*mix router*). Un mezclador es un encaminador que oculta la correspondencia entre mensajes entrantes y salientes a partir de la modificación de su apariencia y del flujo de la transmisión. A continuación veremos cómo se utilizan los mezcladores.

Mezclador

Un mezclador es un encaminador que recibe un conjunto de mensajes de entrada y los devuelve transformados de manera que no se pueda relacionar la entrada con la salida.

Lectura recomendada

El diseño original del mezclador fue propuesto por Chaum en el artículo siguiente:

David L. Chaum (1981). "Untraceable electronic mail, return addresses, and digital pseudonyms". *Communication ACM* (núm. 24, vol. 2, pág. 84-90).

Estas transformaciones se pueden producir tanto en la forma (a fuerza de aplicar técnicas de encriptación y relleno de mensajes) como en la secuencia (a fuerza de mezclar el orden y aplicar diferentes retardos en la entrega de los mensajes).

El diseño original del mezclador se concibió para una red tradicional y consiste en tener un encaminador que procesa los mensajes por lotes. El encaminador almacena mensajes en la memoria hasta que se cumple cierta condición de descarga, momento en el que se envía el lote de mensajes desordenados. La condición de descarga puede ser una condición temporal, espacial o una combinación de ambas. La descarga temporal se establece cada cierto periodo de tiempo (que puede ser fijo o variable), mientras que la espacial se establece cuando se sobrepasa un determinado umbral de capacidad.

Mezclador con piscina

El diseño original del mezclador por lotes se extendió más adelante, de manera que en el momento de la descarga solo se enviase un subconjunto de los mensajes almacenados en el encaminador y el resto se preservase para rondas posteriores. Esta técnica, llamada mezclador con piscina (*pool mix*), mejora el grado de anonimato en situaciones de tráfico

fluctuante compensando un momento de poca carga de tráfico con un mayor retardo en la entrega de los mensajes. Esta solución es ideal para aplicaciones que no tienen restricciones de entrega muy ajustadas, como el correo electrónico anónimo, pero no resulta adecuado para redes que necesitan comunicaciones en tiempo real.

En contraposición con el modelo de mezclador por lotes, tenemos el mezclador continuo, en el que los usuarios definen un retardo aleatorio para cada mensaje e incluyen dicho retardo en la cabecera del mensaje. El mezclador almacena el mensaje durante el tiempo especificado y entonces lo reenvía. La ventaja de este método es que los propios usuarios controlan el tiempo límite de transferencia de la información. Este modelo funciona bien en situaciones de tráfico relativamente estable y constante. Sin embargo, en caso de que se produzcan periodos de tráfico reducido, el grado de anonimato de este modelo es bajo.

Tanto el mezclador continuo como el de piscina son vulnerables a ataques consistentes en la alteración del flujo de $N - 1$ mensajes (N es el número de mensajes umbral necesarios para que se produzca la descarga al encaminador) con el objetivo de trazar un mensaje concreto. Para el caso del mezclador continuo, el atacante debe ser capaz de bloquear la entrada de mensajes al encaminador, mientras que para el mezclador de piscina debería inyectar mensajes marcados que provocasen una descarga controlada del mezclador.

Para mitigar el efecto de este ataque, una solución es utilizar las redes de mezcladores.

Red de mezcladores

Para incrementar el grado de anonimato de un sistema mezclador, los encaminadores mezcladores suelen combinarse formando una red de mezcladores. De este modo, se puede llegar a preservar el anonimato de los usuarios de la red aunque algunos nodos mezcladores resulten comprometidos.

Existen dos topologías básicas de redes de mezcladores:

- **Cascada**: la ruta o rutas que siguen los mensajes están preestablecidas.
- **Ruta libre**: cada mensaje puede seguir una ruta independiente y diferente de los demás mensajes.

Una ventaja de los mezcladores de cascadas sobre los de ruta libre es el hecho de que tienden a concentrar más tráfico en sus rutas, lo que aumenta el grado de anonimato en estas. No obstante, en una cascada un adversario podría llegar a saber exactamente a qué mezcladores ha de controlar para identificar a un usuario en particular. Son sistemas vulnerables a ataques $N - 1$ efectuados por un adversario global. Por otro lado, el establecimiento de una única ruta

debilita la seguridad del sistema resultante al volverlo vulnerable a ataques del tipo *rushing* –en los que el adversario trata de enviar mensajes de descubrimiento de la ruta antes que el nodo fuente para intentar "apropiarse" de la ruta– y a intrusiones de un adversario sobre uno de los nodos de la ruta.

Los modelos de mezcladores combinados tratan de obtener las ventajas de ambas opciones. Un ejemplo es el establecimiento de múltiples cascadas libres. Este caso se basa en el modelo en cascada, pero las rutas se extienden más allá de su destino o bien introducen rutas falsas. Por otro lado, los protocolos de encaminamiento en MANET comienzan a introducir el establecimiento de canales de comunicación entre dos puntos a través de múltiples rutas, lo que los hace más robustos.

2.3.3. Indetectabilidad

Típicamente, las redes anónimas pierden robustez con el transcurso del tiempo, por el hecho de que un análisis exhaustivo de las trazas de la red permite obtener información de los usuarios y de las relaciones entre ellos. Una forma de atacar la raíz de este problema es enmascarar los mensajes entre nodos, de forma que un atacante externo no pueda discernir cuándo la red está enviando datos o cuándo envía ruido.

Entre las técnicas más utilizadas para enmascarar los mensajes, destacan las siguientes:

- **Comunicaciones a ráfagas cortas (*burst communications*)**. La transmisión de mensajes muy cortos es muy difícil de detectar por parte de los usuarios externos a la red. Por eso, este tipo de transmisiones se utiliza para enviar la información de control más sensible de la red.

- **Envío de mensajes ficticios (*dummy data*)**. Su objetivo es conseguir un flujo constante en la red y que el tipo de tráfico (real o falso) sea indistinguible a ojos de un atacante.

- **Modulación por espectro ensanchado (*spread spectrum*)**. Las transmisiones por espectro ensanchado se caracterizan porque la información es enviada a través de un ancho de banda mucho más amplio que el mínimo requerido. Las técnicas más empleadas son los sistemas de secuencia directa y los sistemas de salto de frecuencia. La ventaja de estos sistemas es que la señal es muy difícil de detectar por parte de usuarios que desconozcan la técnica y la codificación utilizadas para la transmisión de la señal.

- **Esteganografía**. Los métodos esteganográficos permiten ocultar un mensaje dentro de cualquier flujo de comunicación de la red, de manera que solo su receptor legítimo pueda extraer la información del canal. Para el resto de los usuarios el mensaje es invisible.

De todas las técnicas para almacenar mensajes, la más sencilla y utilizada es la de envío de mensajes ficticios. Los mensajes ficticios se pueden insertar en la entrada o salida de los mezcladores. Normalmente, la inserción en la salida proporciona mayor anonimato y menor retardo, por el hecho de que el mezclador puede regular de manera más precisa la introducción de mensajes ficticios en la red según el estado del tráfico. Sin embargo, en el caso del ataque $N - 1$, la inserción en la entrada del mezclador puede ofrecer un mayor nivel de protección.

Cuando tratamos redes de mezcladores, los mensajes falsos pueden atravesar varios encaminadores, tal como hacen los demás mensajes. El camino por seguir se determina de manera aleatoria y, normalmente, acaba en el encaminador que lo generó. Esto permite llegar a detectar ataques del tipo $N - 1$ y actuar en consecuencia.

En el entorno de redes *ad hoc*, en las que el uso de recursos es muy limitado, la utilización de este tipo de mensajes debe ser realmente minimizado.

3. Redes de sensores

Una red de sensores inalámbrica puede estar formada por una serie de miles, incluso millones, de sensores (nodos), que poseen capacidad de almacenamiento y procesado y energía limitada. Estas redes se suelen desplegar en entornos hostiles o de difícil acceso para poder obtener regularmente datos del contexto (militar, ambiental, biológico, médico, etc.) y controlarlo. En dichos sistemas, los nodos están expuestos a ataques físicos y de software, por lo que la prevención y el control de la seguridad es un elemento esencial de estudio en las redes de sensores.

Los nodos sensores se pueden considerar como pequeños computadores, extremadamente rudimentarios en términos de características técnicas (número y tipo de interfaces y componentes). Normalmente, consisten en una unidad de procesado (microprocesador) con capacidad computacional limitada y no mucha memoria (lo que sería el sensor propiamente dicho), algún dispositivo de comunicaciones (generalmente radio) y alguna fuente de energía (por lo general, una batería). Algunas implementaciones pueden incluir elementos adicionales, como sistemas de recarga de batería, procesadores secundarios y dispositivos de comunicaciones adicionales (RS-232, USB, etc.). El tamaño del nodo puede variar desde el equivalente a una caja de zapatos hasta dispositivos diminutos como granos de arroz. Las restricciones de tamaño y coste de los nodos sensores afectan en gran medida a la energía, memoria, capacidad computacional y ancho de banda que después podrán utilizar las aplicaciones que requieren su uso.

En este apartado estudiaremos los principales problemas de seguridad relacionados con las redes de sensores. En primer lugar, veremos los esquemas de gestión de claves encargados de la distribución y actualización del material criptográfico en la red para dotar de seguridad a sus comunicaciones. En segundo lugar, veremos cómo se puede optimizar la transferencia de datos desde los sensores hacia una estación base de manera que se pueda garantizar la integridad y autenticidad de los datos.

3.1. Gestión de claves

El establecimiento de claves de seguridad es el servicio básico para poder ofrecer mecanismos de prevención y detección de ataques en redes. Las redes inalámbricas son particularmente vulnerables a escuchas y al hecho de que los nodos pueden ser capturados y comprometidos.

Las soluciones basadas en clave pública son complicadas de desplegar por dos motivos:

1) los sensores poseen capacidades de procesado, almacenaje y fuentes de energía limitadas, el uso de protocolos basados en clave pública como Diffie-Hellman resulta demasiado costoso, y

2) la red no tiene una infraestructura estable y, por tanto, la verificación de revocación de los certificados de clave pública por medio de una autoridad de confianza no se puede llevar a cabo de forma adecuada.

Así pues, se considera que la criptografía de clave asimétrica se debe evitar en las redes de sensores. Las soluciones basadas en claves simétricas son computacionalmente más eficientes, pero más vulnerables a ataques. Los sistemas tienen que asegurar que si un intruso consigue capturar un nodo, no le sea posible acceder a todas las claves de la red y, por tanto, a la información confidencial del sistema.

El objetivo es la creación y gestión de claves simétricas que puedan enfrentarse a todas las necesidades de la red. Los requisitos para el establecimiento de claves dependen de los patrones de comunicación de la red, que son:

1) unidestino, es decir, enviar un mensaje a un único nodo;

2) difusión local, es decir, enviar un mensaje a los nodos vecinos, con los que hay conexión directa, y

3) difusión global, es decir, enviar un mensaje a todos los nodos de la red.

Los mensajes unidestino (*unicast*) se utilizan para enviar información de los datos capturados por un sensor a una estación base o a otro sensor, que desempeñará funciones de agregación de datos. La agregación de datos permite reducir la cantidad de bits transmitidos en una red e incrementa la eficiencia y el tiempo de vida de la red.

Los mensajes de difusión local[15] se utilizan para el control y la gestión de la red, sobre todo para las operaciones de encaminamiento. Los mensajes de difusión global[16] se originan en la estación base y se utilizan para distribuir información de control que involucra a toda la red.

[15] En inglés, *local broadcast*.

[16] En inglés, *global broadcast*.

A partir de estos requisitos de comunicación, se definen cuatro tipos de claves:

- **Clave de nodo**: clave que comparte un nodo y la estación base.
- **Clave de enlace**: clave que comparte un par de nodos vecinos.
- **Clave de grupo**: clave compartida entre un nodo y todos sus vecinos.

- **Clave de red**: clave compartida por todos los nodos de la red y la estación base.

Las claves de los nodos se pueden cargar en los sensores en el proceso de despliegue de una red. La clave de grupo puede ser generada por un nodo y enviada de manera individual a cada uno de sus vecinos protegida con la clave de enlace que comparten los dos. Finalmente, la clave de red también se puede cargar en los nodos antes del despligue de la red. Si se detecta que un nodo ha sido comprometido, sus vecinos tienen que generar nuevas claves de grupo y distribuirlas entre los vecinos honrados. Entonces, la estación base debe generar una nueva clave de red y distribuirla salto a salto por toda la red protegida con las claves de grupo.

Fijaos en que la clave más difícil de generar y gestionar es la clave de enlace entre dos nodos vecinos.

Para generar la clave de enlace, se podría pensar en usar protocolos de criptografía simétrica como Kerberos. En este caso, la estación base desempeñaría el papel de servidor. Para ejecutar el protocolo, se requiere poder enviar mensajes de la estación base a los demás nodos de la red. El problema es que en una red de sensores, el encaminamiento puede depender de otros nodos de la red y, por tanto, puede necesitar que ya estén establecidas las claves de enlace. Además, la carga de comunicación asociada a este protocolo sería muy desigual entre los diferentes sensores de la red. Por último, el esquema sería poco robusto a causa de la presencia de la estación base como punto único de fallo.

Otra aproximación sería precargar las claves de enlace en el sensor antes del despliegue de la red, pero esta opción plantea varios problemas. En primer lugar, en muchas aplicaciones es difícil conocer a priori cuál será el mapa exacto de la red, ya que si, por ejemplo, los sensores se posicionan en su lugar cuando son lanzados por una avioneta, no hay forma de saber con antelación cuáles serán los vecinos de cada nodo. Además, en una red se pueden añadir sensores a posteriori para reemplazar a los sensores defectuosos o que han acabado su ciclo de vida. Es difícil anticipar en tiempo de despliegue de la red dónde se añadirán estos nodos y, por tanto, qué sensores necesitan estar precargados con claves criptográficas adicionales para poder interactuar con los nuevos sensores cuando estos lleguen.

En los siguientes subapartados presentamos diferentes aproximaciones a cómo se puede solucionar el problema del establecimiento de una clave de enlace entre dos nodos vecinos. El primer método se basa en una clave maestra de corto plazo que está presente en cada nodo durante un tiempo limitado después de cada despliegue. Los otros métodos se basan en precargar claves en los sensores de una forma inteligente, sin tener que conocer la topología de la red final y soportando la adición posterior de nuevos nodos en la red.

3.1.1. Establecimiento de claves a partir de una clavemaestra de corto plazo

El establecimeinto de claves de enlace puede aprovechar el hecho de que las redes de sensor son redes relativamente estáticas y formadas por nodos estacionarios. Esto significa que la vecindad de un nodo no cambia a menudo; algunos nodos pueden morir y otros pueden ser añadidos al cabo de un tiempo, pero a pesar de todo, los cambios son muy esporádicos. Por tanto, lo que se puede hacer es utilizar un protocolo de descubrimiento de nodos vecinos y establecer las claves de enlace en el momento del despliegue inicial.

El protocolo consiste en cuatro pasos: carga de la clave maestra, descubrimiento de vecinos, cálculo de la clave de enlace y supresión de la clave maestra. Veamos cada una de estas fases.

La carga de la clave maestra se hace antes del despliegue de la red en un entorno seguro. Durante esta fase, la clave maestra K_{init} se carga en los nodos, y cada nodo u calcula una clave maestra del nodo llamada K_u, tal que $K_u = f_{K_{init}}(u)$, donde f es una función seudoaleatoria.

La fase de descubrimiento comienza justo después del despliegue de un nodo. En primer lugar, el nodo inicializa un temporizador con un tiempo T_{min}. Después intenta descubrir a sus vecinos enviando un mensaje *hello* en difusión que contiene su identificador. Un nodo vecino v que escuche el mensaje *hello* responde con un mensaje *ack* en el que envía su identificador v. El mensaje *ack* también contiene un código de autenticación MAC generado con la clave maestra del nodo v, K_v. Dado que el nodo u todavía posee la clave maestra K_{init}, puede calcular $K_u K_u$ y verificar el MAC.

Una vez finalizada la fase de descubrimiento de nodos, el nodo u entra en la fase de cálculo de la clave de enlace. La clave de enlace K_{uv} entre los nodos u y v se calcula como $K_{uv} = f_{K_v}(u)$. Observamos que los dos nodos pueden calcular esta clave sin necesidad de intercambiar ningún mensaje más. El nodo u no ha sido autenticado explícitamente por el nodo v. Sin embargo, todos los mensajes que u envíe a v serán autenticados con la clave K_{uv} y, por tanto, se puede asegurar que el nodo será siempre el mismo.

Finalmente, cuando el tiempo del temporizador T_{min} finaliza, el nodo u ejecuta la fase de supresión de la clave maestra borrando K_{init} y todas las claves maestra de nodos K_v de su memoria. Sin embargo, no borra su propia clave maestra K_u, y esta se utilizará más adelante para establecer claves de enlace con nodos que se añaden posteriormente a la red.

3.1.2. Piscinas de claves

Este tipo de esquemas se basa en una predistribución aleatoria de claves lo más óptima y eficiente posible. En un principio, se genera un amplio conjunto de claves, lo que llamamos piscina de claves. Antes de desplegar la red, se entrega a cada nodo un subcojunto aleatorio de la piscina de claves, un anillo de claves. El hecho de que los nodos solo tengan que almacenar un anillo permite reducir los requisitos de memoria y hace que el sistema sea escalable y apropiado para sensores. Del mismo modo, es evidente que inicialmente no todos los pares de vecinos de la red comparten una clave. Veremos que los protocolos basados en piscina permiten que cualquiera de los dos nodos que al principio no comparten una clave puedan establecer otra con una alta probabilidad, a través de la comunicación con nodos intermedios.

Lectura recomendada

El primer trabajo que utiliza el concepto de piscina de claves es:

Laurent Eschenauer; Virgil D. Gligor (2002). "A key-management scheme for distributed sensor networks". En: *Proceedings of the 9th ACM conference on Computer and communications security, CCS '02* (pág. 41-47). ACM: Nueva York.

Para calcular la probabilidad de que dos nodos de la red comparten una clave, expresamos el problema de la manera siguiente: dado un conjunto S de k elementos, elegimos de forma aleatoria dos subconjuntos, S_1 y S_2, de m_1 y m_2 elementos cada uno. La probabilidad de $S_1 \cap S_2 \neq 0$ es:

$$P\left\{S_1 \cap S_2 \neq 0\right\} = 1 - \frac{(k-m_1)!(k-m_2)!}{k!(k-m_1-m_2)!} \qquad 6.1$$

Cuando k es muy grande, podemos utilizar la aproximación de Stirling para $n!$,

$$n! \approx \sqrt{2\pi}\, n^{n+\frac{1}{2}} e^{-n} \qquad 6.2$$

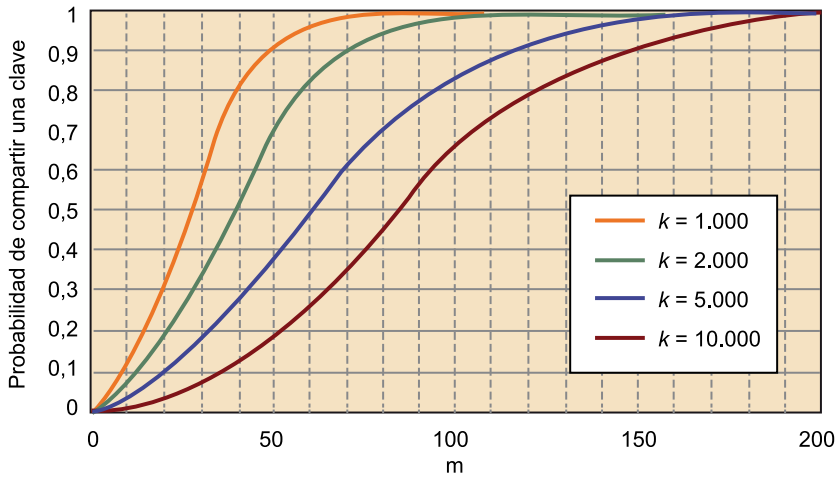
y, por tanto, asumiendo que $m = m_1 = m_2$ y simplificando la expresión 6.1, obtenemos:

$$P\left\{S_1 \cap S_2 \neq 0\right\} = 1 - \frac{\left(1-\frac{m}{k}\right)^{2(k-m+\frac{1}{2})}}{\left(1-\frac{2m}{k}\right)^{(k-2m+\frac{1}{2})}} \qquad 6.3$$

Ejemplo

La figura 8 ilustra los valores de la expresión 6.3 para k = 1.000, k = 2.000, k = 5.000 y k = 10.000. Como se puede observar, la probabilidad de que dos usuarios compartan una clave crece rápidamente con m. Por ejemplo, para k = 1.000 claves, solo hay que distribuir grupos de 26 claves entre los nodos para que la probabilidad de que dos vecinos compartan una clave sea mayor del 50%. Si incrementamos en orden de magnitud la dimensión de la piscina de claves (k = 10.000), el número de claves que se tienen que distribuir entre los usuarios para tener una $P > 50\%$ es m = 83, que solo es 3,2 veces el número de claves distribuidas en el caso de una piscina k = 1.000. Así pues, la solución de la piscina de claves es bastante escalable.

Figura 8. Predistribución de claves aleatoria



Durante la fase de inicialización de los protocolos basados en piscina de claves, se establecen las medidas de la piscina y del anillo de claves, de modo que sea fácil encontrar a dos vecinos que compartan una clave pero sea difícil encontrar a tres o más que también lo hagan. A partir de aquí, se genera la piscina y se va asignando de manera aleatoria un anillo de claves a cada sensor.

Cuando todos los sensores ya están desplegados, se inicia la fase de establecimiento directo de una clave. En esta fase, los sensores llevan a cabo un descubrimiento de los nodos vecinos con los que comparten una clave común. El descubrimiento de claves se puede implementar asignando un identificador corto a cada clave de S antes del despliegue y haciendo que cada nodo envíe en difusión el conjunto de identificadores correspondiente a las claves de su anillo. Cuando dos vecinos descubren que comparten una clave, pueden verificar que ambos tienen la clave ejecutanto un protocolo de reto-respuesta. La clave que comparten se establece entonces como la clave de enlace entre ellos.

Algunos pares de vecinos pueden no tener ninguna clave en común. Si esto sucede, lo que hacen es ejecutar la fase de establecimiento multisalto de una clave en la que los nodos acuerdan una clave compartida a través de una ruta formada por nodos de la red que ya tienen una clave de enlace definida y que permiten unir el par de sensores. La clave compartida viene definida por uno de los dos nodos a partir de una clave no utilizada de su anillo de claves.

La principal ventaja de este esquema es que los sensores no tienen que hacer cálculos intensivos, es escalable y es fácil añadir nodos al sistema. El problema es que si un nodo es capturado y comprometido, sus claves pueden ser utilizadas por otros nodos. Por otro lado, si el nodo comprometido participa en el establecimiento multisalto de una clave para un nodo vecino, la clave del vecino también resulta comprometida. Finalmente, este sistema no proporciona autenticación nodo a nodo. Esto significa que un nodo puede establecer unas

claves compartidas con sus vecinos, pero no sabe exactamente quiénes son dichos vecinos. Así pues, la expulsión de nodos maliciosos o comprometidos de la red no es posible.

Predistribución de claves *q*-compuesta

Una forma para mejorar la capacidad de recuperación del esquema básico de la piscina de claves ante ataques de captura de nodos es utilizar una predistribución aleatoria de claves *q*-compuesta. Este esquema se diferencia del esquema básico en que los nodos han de tener como mínimo q claves en común en sus anillos de claves para poder establecer una clave de enlace. La clave de enlace se calcula como el resumen de todas las claves compartidas.

Este esquema es más robusto al compromiso de una clave de enlace, ya que para ello el atacante debe ser capaz de capturar un nodo y descubrir qué subgrupo del anillo de claves se está utilizando en cada enlace. Por otro lado, los requisitos del protocolo también son mayores, ya que la probabilidad de establecer una clave directamente como otro usuario es menor que en el esquema de la piscina de claves (es menos probable tener q claves comunes con un vecino que tener solo una), y para solventarlo se puede incrementar el tamaño del anillo de claves (con la necesidad consiguiente de más memoria) o el tamaño de la piscina de claves ha de ser más pequeño (con el efecto negativo que esto tendría sobre la captura de nodos).

Nota

Cuando $q = 1$, el esquema *q*-compuesto tiene el mismo comportamiento que el esquema básico de la piscina de claves.

Refuerzo de la clave a través de multicamino

Otra alternativa de mejora en el esquema básico de la piscina de claves es establecer las claves por medio de caminos disyuntos. Este protocolo se aplica cuando ya se ha podido establecer una clave de enlace directa k_0 a través de la piscina. Entonces, uno de los nodos identifica un conjunto de j caminos disyuntos hacia el otro nodo y le envía j claves compartidas $k_1,k_2,...,k_j$, una por cada camino. Cada clave está protegida salto a salto con las claves de enlace de los nodos que colaboran en la retransmisión. Cuando el nodo receptor recibe las claves, ambos calculan la clave final de enlace como $K = k_0 \oplus k_1 \oplus \cdots \oplus k_j$.

Este sistema comporta una mayor sobrecarga para el establecimiento de claves, pero tiene la ventaja de que comprometer una clave de enlace resulta más costoso. Para hacerlo, un atacante necesitaría comprometer como mínimo una clave de enlace de cada una de las rutas que participan en el establecimiento de claves.

3.1.3. Predistribución aleatoria de claves basada enpolinomios

El objetivo de este esquema es minimizar la información sensible que un atacante obtiene cuando captura un sensor. La idea es que no sea posible obtener información útil si no se captura un número mínimo de nodos, y por eso se utiliza la criptografía umbral.

A continuación describimos cómo funciona la predistribución de claves basada en polinomios y después mostramos cómo se puede combinar con una piscina de claves:

1) Escogemos un polinomio bivariado de grado t sobre un cuerpo finito $GF(p)$, donde p es un número primo grande tal que $f(x, y) = f(y, x)$.

$$f(x, y) = \sum_{i,j=0}^{t} a_{ij} x^i y^j \qquad 6.4$$

2) Cada sensor es precargado con un fragmento del polinomio $f(id_i, y)$, donde id_i es el identificador del sensor i.

3) Cualquiera de los dos nodos i, j puede calcular una clave compartida. Para ello:

- i evalúa su fragmento de polinomio en el punto id_j, y obtiene $f(id_i,id_j)$.
- j evalúa su fragmento de polinomio en el punto id_i, y obtiene $f(id_j,id_i)$.

Este esquema es incondicionalmente seguro y resistente a ataques de t coaliciones, es decir, cualquier ataque que comprometa hasta t nodos no puede obtener ninguna información sobre las claves compartidas calculadas por cualquier par de los nodos comprometidos. El esquema no presenta sobrecarga de comunicaciones para los sensores, pero sí que tiene requisitos de memoria relevantes, en concreto $(t + 1)log(p)$ por nodo. Esto hace que el umbral t esté limitado por la capacidad de memoria de los sensores.

Ejemplo

Se diseña la distribución de claves para una red de sensores que pueda soportar ataques de hasta tres nodos de coalición. Se elige el polinomio de trabajo siguiente en GF(139): $f(x, y) = 3 + 5xy + 18x^2y^2 + 2x^3y^3$. Los sensores id_{24} e id_{15} reciben los siguientes fragmentos del polinomio:

- Sensor id_{24}: $f(y) = 3 + 120y + 82y^2 + 126y^3$.
- Sensor id_{15}: $f(y) = 3 + 75y + 19y^2 + 78y^3$.

Una vez desplegada la red de sensores, los nodos id_{24} e id_{15} son vecinos y quieren establecer una clave de enlace. Entonces lo que hacen es:

- Sensor id_{24}: $f(15) = 3 + 120.24 + 82.24^2 + 126.24^3 = 8$.

- Sensor id_{15}: $f(24) = 3 + 75.15 + 19.15^2 + 78.15^3 = 8$.

La clave compartida es 8.

Para poder ofrecer un esquema más robusto ante atacantes sin incrementar los requisitos de memoria, lo que se hace es combinar la idea de la predistribución de claves basada en polinomios con la piscina de claves. Es lo que llamamos esquemas de predistribución aleatoria de claves basada en polinomios. Su funcionamiento es el siguiente:

1) Generamos una piscina *S* de polinomios bivariados de grado *t*.

2) Para cada sensor *i* escogemos un subconjunto de *m* polinomios de la piscina, y precargamos el sensor con un fragmento de cada polinomio calculado en el punto id_i.

3) Dos nodos que tengan fragmentos de un mismo polinomio *f* podrán establecer una clave compartida *f*(*i*,*j*).

4) Si dos nodos no tienen polinomios en común, pueden establecer una clave compartida a través de un camino de intermediarios.

Como en el esquema anterior, si un atacante compromete un nodo, puede obtener información de las claves de enlace que dicho nodo utiliza con sus vecinos, pero en ningún caso obtendrá información reveladora sobre las claves compartidas de cualquier otro par de claves. Para comprometer un polinomio, un adversario necesitaría obtener $t + 1$ fragmentos de dicho polinomio. Conseguir estos $t + 1$ fragmentos no resulta fácil, ya que todos los nodos tienen fragmentos de polinomios diferentes de la piscina y, por tanto, se tienen que capturar muchos nodos para poder conseguir $t + 1$ fragmentos de un solo polinomio. Eso sí, cuando se consigue comprometer un polinomio, todos los pares de nodos que utilicen fragmentos de este se verán afectados.

Los requisitos de memoria de este esquema son $m(t+1)log(p)$. Vemos que se diferencia del esquema de predistribución de claves basada en polinomios en solo un factor *m*, pero en cambio puede soportar muchos más ataques y, por tanto, el valor de *t* puede ser mucho menor.

3.1.4. Predistribución de claves basada en matrices

Los esquemas de predistribución de claves basada en matrices se fundamentan en el esquema de Blom propuesto en 1985. En este esquema, una matriz simétrica $K_{n\times n}$ guarda todas las claves de un grupo de *n* nodos, donde cada elemento k_{ij} es la clave de enlace entre los nodos *i* y *j*. El esquema de Blom puede resistir ataques de hasta *t* nodos. Todas las claves de enlace de los nodos no comprometidos de una red se mantienen seguras mientras no haya más de *t* nodos corruptos.

La predistribución de claves de Blom funciona de la forma siguiente:

1) Generamos una matriz $G_{(t+1)\times n}$ sobre un cuerpo finito $GF(p)$, donde n es la dimensión de la red y p es un número primo mayor que la longitud de la clave deseada y mayor que n. G es una matriz pública y puede ser compartida por diferentes sistemas.

2) La estación base de la red de sensores genera una matriz simétrica aleatoria $D_{(t+1)\times(t+1)}$ sobre el cuerpo finito $GF(p)$. D es una matriz secreta.

3) A partir de las matrices secreta y pública, la estación base calcula una matriz $A_{n\times(t+1)}$ y $K_{n\times n}$:

a) $A = (DG)^T$.

b) $K = AG$. Podemos comprobar que K es una matriz simétrica, ya que $K = AG = (DG)^T G = G^T D^T G = G^T DG = G^T A^T = (AG)^T = K^T$.

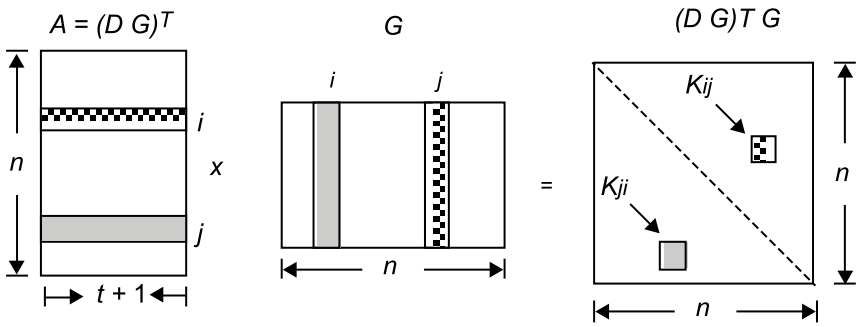
4) Cada nodo i guarda la fila i de la matriz A.

Una vez desplegada la red, cualquiera de los dos nodos i y j pueden calcular una clave compartida K_{ij} intercambiando sus columnas de la matriz G con texto llano y haciendo:

- i calcula $A(i,.)G(.,j) = K_{ij}$.
- j calcula $A(j,.)G(.,i) = K_{ji} = K_{ij}$.

Se observa que la clave es el producto de una fila de la matriz privada y una columna de la matriz pública, y dado que las filas de A siempre se mantienen en secreto, un observador que capture el tráfico entre los nodos i y j no podrá deducir la clave de enlace que establecen entre ellos (ved la figura 9).

Figura 9. Generación de claves en el esquema de Blom



Se puede demostrar que el esquema de Blom es *t*-seguro si las $t+1$ columnas de la matriz G son linealmente independientes. Una posible manera de construir la matriz G es utilizando una matriz de Vandermonde de la forma siguiente:

$$G=\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ s & s^2 & s^3 & \cdots & s^n \\ s^2 & (s^2)^2 & (s^3)^2 & \cdots & (s^n)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s^t & (s^2)^t & (s^3)^t & \cdots & (s^n)^t \end{bmatrix} \qquad 6.5$$

Ejemplo

Tenemos una red de sensores con $n = 5$ nodos que utiliza una predistribución de claves de Blom con los siguientes datos:

$$p=17,\quad t=3,\quad G=\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 \\ 1 & 3 & 9 & 10 & 13 \\ 1 & 4 & 16 & 13 & 1 \end{bmatrix},\quad D=\begin{bmatrix} 1 & 6 & 2 & 1 \\ 6 & 3 & 8 & 10 \\ 2 & 8 & 2 & 13 \\ 1 & 10 & 13 & 4 \end{bmatrix} \qquad 6.6$$

Entonces, la matriz privada de claves es:

$$A=(DG)^T mod 17=\begin{bmatrix} 10 & 10 & 8 & 11 \\ 6 & 8 & 8 & 8 \\ 8 & 12 & 5 & 1 \\ 14 & 2 & 0 & 8 \\ 5 & 15 & 16 & 11 \end{bmatrix} \qquad 6.7$$

Si los usuarios 2 y 4 quieren establecer una clave de enlace entre ellos:

1) El usuario 2 calcula k_{24} con su fila privada de A, y la matriz pública G, de manera que:

$$[6\ \ 8\ \ 8\ \ 8]\begin{bmatrix} 1 \\ 8 \\ 10 \\ 13 \end{bmatrix} mod 17=16 \qquad 6.8$$

2) El usuario 4 calcula k_{42} con su fila privada de A, y la matriz pública G, de manera que:

$$[14\ \ 2\ \ 0\ \ 8]\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} mod 17=16 \qquad 6.9$$

Para poder ofrecer un esquema más robusto ante atacantes, podemos combinar el esquema de Blom con la piscina de claves, con lo que obtenemos un esquema de predistribución aleatoria de claves basada en matrices. Su funcionamiento es el siguiente:

1) Generamos una matriz pública $G_{(t+1)\times n}$ sobre un cuerpo finito $GF(p)$ como en el caso anterior.

2) Generamos k matrices simétricas aleatorias $D_{(t+1)\times(t+1)}$ sobre el cuerpo finito $GF(p)$.

3) A partir de G y las k matrices D, obtenemos k matrices $A_{n\times(t+1)}$ que conforman el contenido de la piscina: $A_V = (D_V G)^T$.

4) Para cada sensor i escogemos un subconjunto de m matrices A de la piscina, y precargamos en el sensor la fila i de las matrices seleccionadas (por ejemplo, $A_V(i,.)$ para cada V seleccionada).

5) Dos nodos que tengan seleccionada una matriz común A_V pueden calcular una clave compartida utilizando el esquema de Blom.

6) Si dos nodos no tienen una matriz en común, pueden establecer una clave compartida a través de un camino de intermediarios.

3.2. Agregación de datos

Los datos recogidos por los sensores se tienen que procesar de alguna manera para poder extraerles información interesante. En muchas ocasiones no es preciso enviar todos los datos recogidos por los sensores (uno a uno) a la aplicación final, sino que los datos se pueden presentar en un único punto de vista que sintetice todo lo recogido por los sensores (suma de todos los datos, media, etc.).

La agregación de datos es una forma eficiente de ahorrar energía a los sensores. Del mismo modo, también crea un conjunto de nuevos riesgos de seguridad:

1) Si los datos tienen que viajar a través de una red multisaltos, los nodos que reenvían la información de los sensores pueden manipular los datos.

2) Se pierde la identificación de las fuentes de información, con lo que la detección de nodos maliciosos resulta más difícil.

3) El nodo agregador puede falsificar el resultado de la agregación de datos.

Para evitar estos problemas, deberíamos proporcionar servicios de integridad y autenticidad de datos desde que salen del sensor hasta que llegan a la aplicación final. Generalmente, esto se podría conseguir con códigos de autenticación de mensajes o con firmas digitales, pero ambas opciones son demasiado costosas para ponerlas en práctica con todos los datos que salen de los sensores. En las redes de sensores, todas las operaciones tienen que ser muy eficientes, dadas las limitaciones de los nodos.

En este subapartado veremos cómo los métodos basados en funciones resumen proporcionan firmas eficientes en entornos muy limitados.

3.2.1. Árbol de Merkle

Un árbol de Merkle es un árbol de resúmenes que permiten la preautenticación de un conjunto de valores con una sola firma digital (como en el caso de las cadenas resumen).

Ralph Merkle

El árbol de Merkle es una construcción introducida por Ralph Merkle en 1979 con el objetivo principal de hacer más eficiente el manejo de múltiples firmas de un solo uso.

El árbol de Merkle es un árbol en cuyas hojas se encuentran valores de resumen de bloques de datos de, por ejemplo, los valores obtenidos por un sensor.

El esquema del árbol de Merkle funciona de la manera siguiente. Tomemos $x_1,x_2,\ldots,x_{2^l}$ como los valores que queremos autenticar. En primer lugar calculamos un resumen de estos valores con una función $h:\{0,1\}^* \rightarrow \{0,1\}^n$, de manera que obtenemos $y_1,y_2,\ldots,y_{2^l}$, donde $y_i = h(x_i)$. Entonces asignamos los valores y_i a las hojas de un árbol binario. Además, en cada vértice interno de este árbol asignamos un valor que se calcula como el resumen de los valores asignados a sus dos hijos. Finalmente, se firma el valor asignado a la raíz del árbol.

Cuando el propietario del árbol quiere autenticar alguno de los valores x_i, revela el valor x_i y todos los valores asignados en los vértices hermanos en el camino entre y_i y la raíz del árbol. El verificador puede hacer un resumen de estos valores en el orden apropiado y comparar el resultado con el valor asignado a la raíz (y del cual tiene una firma digital). Si los dos valores concuerdan, entonces el valor x_i es auténtico (es decir, se verifica que el valor proviene de la misma entidad que calculó el árbol y firmó digitalmente la raíz).

Dada la propiedad de unidireccionalidad de las funciones resumen, el hecho de revelar un valor x_i y los valores asignados a los nodos hermanos no permite calcular ningún otro valor de x_j, $i \neq j$ del árbol. Observad que esto provoca que los valores de un árbol de Merkel se puedan revelar en cualquier orden, a diferencia de lo que sucede en una cadena resumen.

Figura 10. Árbol de Merkle de nivel $K = 4$

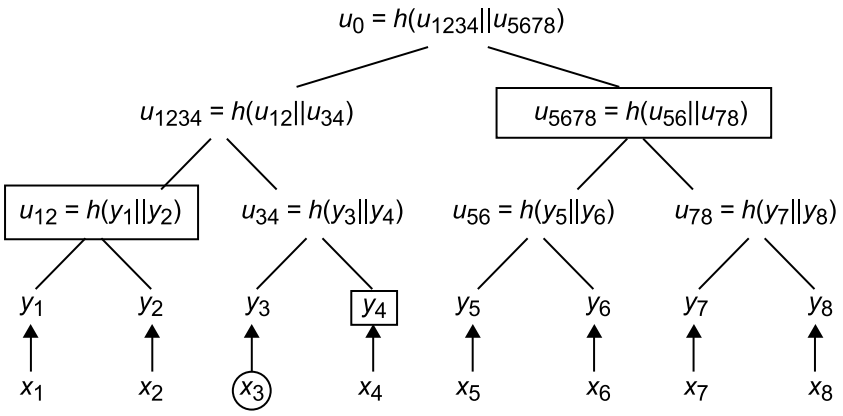


Figura 10

La figura muestra que el valor x_3 está autenticado revelando y_4, u_{12} y u_{5678}. El verificador puede calcular $h(h(u_{12} \| h(h(x_3) \| y_4)) \| u_{5678})$ y comparar el resultado con el valor u_0 asignado a la raíz del árbol. Si coinciden, x_3 está autenticado favorablemente.

En el caso de una red de sensores, supongamos que los sensores se asocian de forma jerárquica. Cada sensor genera una lectura de datos y un resumen de la lectura. Los datos y el resumen se pasan al nodo padre. El padre genera un resumen de todos los resúmenes que ha recibido de sus hijos y lo envía al siguiente nodo del árbol. El proceso continúa hasta que el nodo raíz obtiene un resumen que agrega todos los datos.

En cualquier momento, la estación base puede pedir a algún sensor que le envíe la lectura de datos de un instante determinado para comprobar que los datos que va recibiendo de los nodos recolectores son correctos.

4. Protocolos tolerantes a retardos e interrupciones

La arquitectura y protocolos actuales de Internet han demostrado ser perfectamente adecuados para una gran variedad de aplicaciones, pero siempre y cuando haya un buen nivel de conectividad entre las partes en comunicación. En cambio, el rendimiento de estos protocolos se puede degradar fácilmente, o incluso pueden dejar de funcionar por completo, en escenarios en los que no exista un canal continuo entre las partes que se comunican, o en los que el retardo de las comunicaciones sea significativo. Dos ejemplos de estas situaciones degradadas se dan cuando usuarios de dispositivos móviles intentan comunicarse mientras están situados en redes inalámbricas *ad hoc* diferentes o cuando las aplicaciones que intentan comunicarse no están conectadas de manera simultánea o tienen un acceso temporal discontinuo a redes de comunicación. En el primer caso hablamos de una conectividad intermitente en el espacio y en el segundo, de una conectividad intermitente en el tiempo.

La solución para los escenarios de conectividad intermitente que prevé un enfoque más global y que más éxito ha tenido hasta el momento es la basada en los protocolos tolerantes a retardos e interrupciones (*delay and disruption tolerant networking*, DTN) (3; 2).

En el enfoque DTN, las comunicaciones se realizan exclusivamente en el nivel de aplicación, entre nodos adyacentes cuando estos tienen la oportunidad de comunicarse, y sin necesidad de una conectividad continua de extremo a extremo. Este método permite las comunicaciones incluso en las situaciones más adversas, pero a costa de perder la interactividad, ya que cada paso de una comunicación puede llevar un tiempo no limitado.

La red DTN de la NASA

El ejemplo más mediático de protocolos DTN, aunque sin duda uno de los menos cotidianos, es el de la comunicación entre sondas enviadas al espacio exterior y la posterior retransmisión de datos a las bases terrestres (por ejemplo, en las últimas misiones a Marte). La NASA hizo público a finales del 2008 que probó con éxito la primera red de comunicaciones en el espacio exterior modelada según los principios de Internet, pero utilizando protocolos DTN. En estas situaciones de distancias ingentes, el tiempo de propagación de los mensajes es muy superior a los máximos establecidos por TCP y, por tanto, hasta ahora las comunicaciones interplanetarias tenían que ser modeladas de manera específica para cada misión, sin poder aprovechar el enorme potencial que comporta la experiencia de la comunidad Internet.

Enlace de interés

Podéis acceder a los detalles de la experiencia de la NASA en el uso de una red DTN en la dirección http://www.jpl.nasa.gov/news/news.cfm?release=2008-216.

Existen muchos otros escenarios más "terrestres" en los que los protocolos DTN pueden ser de utilidad. De hecho, tales escenarios de conectividad intermitente son cada vez más frecuentes a medida que las redes inalámbricas y los dispositivos móviles se vuelven más omnipresentes en los entornos urbanos de nuestra sociedad. También son más habituales a medida que queremos extender la cobertura de Internet hacia entornos no considerados anteriormente,

como áreas rurales aisladas o de países subdesarrollados, sin (buenas) infraestructuras de comunicación; hacia áreas de países árticos, sin cobertura de satélite; o con dispositivos diminutos como los sensores (la llamada Internet de las cosas) con limitada capacidad comunicativa. En algunos casos, las aplicaciones buscan expresamente ser independientes de las infraestructuras convencionales de comunicación (GSM/GPRS/UTM/WiMAX), tanto para ahorrarse los costes de comunicación asociados como para superar su modelo inherente de comunicaciones continuas entre los extremos.

Existen diferentes tipos de DTN asociados a las características de las aplicaciones y de los entornos de comunicación. En el caso de satélites y sondas espaciales, las trayectorias son predecibles y, por tanto, se puede saber por adelantado en qué momentos es posible la conexión. En cambio, si consideramos una red de sensores instalados alrededor de un lago para controlar su nivel de contaminación, no se puede predecir en qué instantes estos sensores quedarán cubiertos por el agua o sin posibilidad de cargar la batería por culpa del mal tiempo, y por lo tanto, sin posibilidad de comunicarse. En consecuencia, para cada situación habrá que identificar una aproximación al concepto DTN que se ajuste a sus características: móviles o estáticos, con disponibilidad continua o interrumpida, etc.

Lectura complementaria

S. Farrell; V. Cahill (2006). *Delay- and Disruption- Tolerant Networking*. Artech House. El capítulo 3 de este libro contiene una descripción detallada de una docena de aplicaciones DTN.

En la IRTF (Internet Research Task Force) hay un grupo de investigación dedicado a los protocolos DTN. Las actividades del DTNRG (Delay and Disruption Tolerant Networking Research Group) han consistido en la publicación de borradores de dos propuestas de protocolos (*bundle protocol*, BP, y *licklider transmission protocol*, LTP), en el desarrollo de implementaciones de dichos protocolos y en la publicación de hasta once estándares de Internet (*Request for Comments*, RFC) de carácter informativo y experimental desde el 2007 hasta el 2011. El protocolo BP está pensado para necesidades DTN terrestres y el LTP, para necesidades DTN en el espacio exterior.

Enlace de interés

Podéis acceder a las actividades del grupo de investigación en DTN de la IRTF en la dirección http://www.dtnrg.org.

Dado que las situaciones de conectividad intermitente pueden ser muy diversas, existen otras propuestas de arquitecturas de protocolos DTN que no siguen el modelo DTNRG. Los siguientes subapartados describen las arquitecturas DTN más significativas: la arquitectura del DTNRG, la arquitectura colapsada de Haggle y la arquitectura basada en mensaje activos.

4.1. Arquitectura DTN propuesta por el DTNRG

La arquitectura "oficial" para los protocolos DTN se define en el RFC 4838 (2). Este estándar sitúa las funcionalidades necesarias para la tolerancia a retardos e interrupciones en una capa propia integrada en la capa de aplicación, o como mínimo por encima de la capa de transporte. En este sentido, el enfoque DTN representa una capa superpuesta (*overlay network*) que puede situarse por encima de cualquier combinación de redes existentes.

RFC4838

El estándar de Internet (*Request for Comments*, RFC) con numeración AAAA se puede descargar de http://www.rfc-editor.org/rfc/rfcAAAA.txt.

Podemos observar esta arquitectura en la figura 11*b*. En esta figura también se muestra cómo se hace la transferencia de datos en el modelo DTN, comparada con una transferencia de datos a la Internet actual, que se muestra en la figura 11*a*.

Figura 11. Arquitectura de protocolos DTN

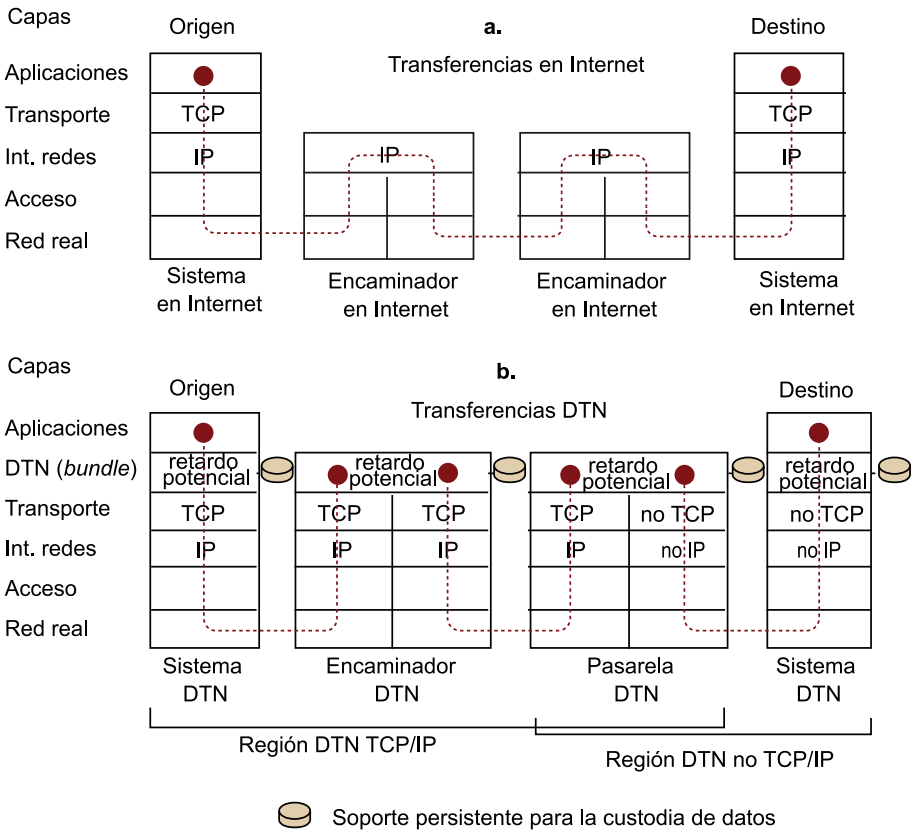


Figura 11

a. Esquema de comunicación tradicional de Internet.

b. Esquema de comunicación DTN, con la nueva capa DTN entre las capas de transporte y de aplicaciones.

En Internet (figura 11*a*) se encuentra una serie de encaminadores o *routers* que interconectan un conjunto de redes diferentes hasta ofrecer a las aplicaciones una visión global y única. Dichas aplicaciones disponen de una comunicación de extremo a extremo (por medio de conexiones TCP o de datagramas de usuario UDP) que les permite la comunicación directa, permanente y continua. Los datagramas IP salen de la máquina de un extremo y van pasando por los encaminadores intermedios hasta llegar a la máquina del otro extremo.

En un entorno DTN (figura 11*b*), al no haber conectividad de extremo a extremo, la capa DTN proporciona una conectividad intermitente entre las diferentes redes subyacentes. Fijaos en que estas redes subyacentes pueden llegar a ser nodos individuales, donde cada nodo se comporta como encaminador o pasarela DTN. La capa DTN recoge los datos de una aplicación y los agrupa hasta formar una unidad de datos llamada paquete (*bundle*), que será almacenada dentro de la capa en un soporte persistente y no se enviará hacia el nodo o encaminador siguiente hasta que la comunicación sea posible. En esta co-

municación no hay restricciones temporales ni restricciones de conectividad de ningún tipo. Se dice que cada nodo actúa como custodio de los paquetes de datos que están en tránsito desde el nodo origen hasta el nodo destino.

La capa DTN proporciona un mecanismo de interconexión de redes similar al ofrecido por la capa IP, pero en el nivel de de la aplicación. Las capas inferiores a la DTN no tienen que ser necesariamente las capas de Internet (TCP/IP) tal como las conocemos, sino que pueden ser las capas de cualquier familia de protocolos. En la figura 11*b* se distingue entre un encaminador DTN, que interconecta dos redes con protocolos TCP/IP (*internets*) que, además, tienen la capa DTN, y una pasarela DTN, que interconecta una *internet* con una red con protocolos no TCP/IP. Las funciones de custodia de los paquetes de datos son las mismas tanto en los encaminadores como en las pasarelas.

4.2. Arquitectura colapsada Haggle

Enlace de interés

Se puede acceder a toda la información y resultados del proyecto europeo Haggle en http://www.haggleproject.org/.

Haggle fue un proyecto europeo, finalizado en el 2010, que se proponía ofrecer aplicaciones para la comunicación intermitente entre dispositivos inteligentes sin utilizar ningún tipo de infraestructura. Se dice que la comunicación es oportunista porque se asume que solo es posible cuando los dispositivos entran en la cobertura de la red inalámbrica que incorporan, es decir, los dispositivos se comunican solo cuando tienen la oportunidad de hacerlo y no se preocupan de las desconexiones. Un ejemplo de ello sería la comunicación que se puede establecer entre dos *smartphones* de dos personas que coinciden unos instantes en el andén de una estación.

Haggle prevé una arquitectura de tres capas (ved la figura 12):

- una capa superior de las aplicaciones y una capa inferior de conectividad a través de cualquier tecnología inalámbrica disponible;

- entre la capa de conectividad y la de aplicaciones no hay ninguna capa tradicional TCP/IP, sino una sola capa (por eso se denomina arquitectura colapsada) que proporciona toda la funcionalidad para recibir un paquete de datos o más de uno en una oportunidad de comunicación, custodiarlos, encaminarlos y reenviarlos cuando se tenga una nueva oportunidad de comunicación.

Figura 12. Arquitectura DTN de Haggle

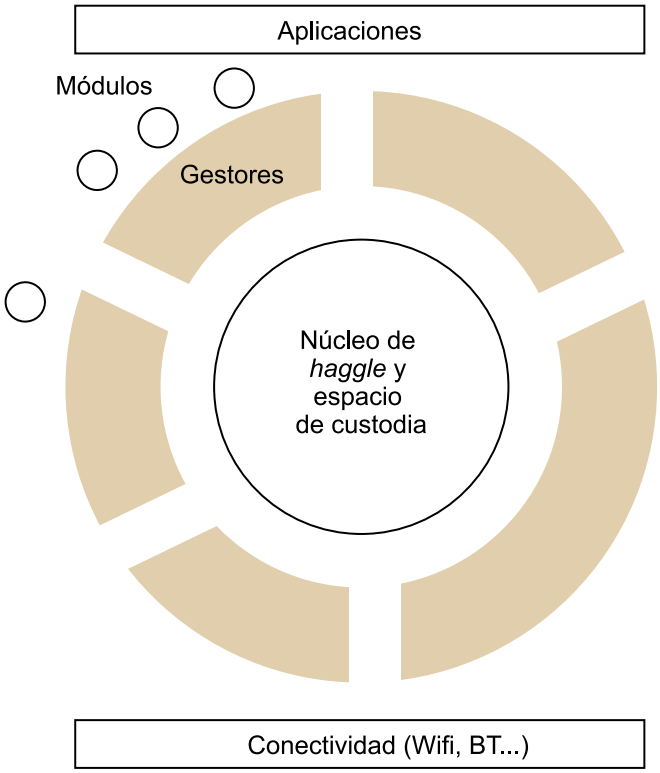


Figura 12

La capa Haggle intermedia trabaja directamente sobre la red inalámbrica inferior y por debajo de las aplicaciones. Contiene una serie de gestores y módulos para recibir, almacenar y reenviar los datos de una manera oportunista.

4.3. Arquitectura DTN con mensajes activos

Hay una gran dependencia entre la aplicación concreta que utilizará la red DTN y el algoritmo de encaminamiento. Esto no se ve en redes muy conectadas, como por ejemplo Internet, ya que el mejor algoritmo para encaminar, es decir, para decidir cuál es el mejor vecino para transmitirle un mensaje con el fin de que llegue a su destino es único y conocido. Todos los encaminadores de Internet siguen el mismo algoritmo, descrito en el estándar del protocolo IP, que va bien para cualquier aplicación.

Sin embargo, en el caso de las DTN, la situación es muy diferente. Ya no se tiene una única red conectada, sino que habrá momentos de discontinuidad en los que el mensaje deberá ser retenido. Cuando llega un conjunto de vecinos a este nodo, hay que decidir cuál es la mejor opción, y aquí estriba una de las mayores dificultades de las DTN. Dado que hay muchas posibilidades de tener redes no conectadas continuamente, no podemos establecer cuál será la estrategia que irá mejor siempre. Por otro lado, cada aplicación puede tener necesidades de encaminamiento diferentes, y puede disponer de información adicional que permitiría hacer una mejor elección del siguiente nodo para que el mensaje llegue a su destino de la mejor manera posible.

Algunos ejemplos de DTN

SENDT: monitorización de la contaminación de lagos. En este proyecto se utilizan sensores sin mantenimiento para medir la contaminación en lagos irlandeses. Las barcas de los pescadores sirven de mulas de datos, recolectan toda la información de los sensores y la llevan a tierra (13).

Zebranet: seguimiento de zebras en Kenya. Las zebras llevan un collar con un receptor GPS que va almacenando su posición cada pocos minutos. Cada dos horas se intenta enviar toda la información a alguna zebra cercana para que la retransmita a otra, y finalmente llegue a un colector de datos (17).

Sapo gigante en Australia. Para monitorizar la invasión del sapo gigante en Australia, se utilizan unos sensores que detectan el canto de esta especie. Unas mulas de datos recogen la información recopilada por estos sensores y la transportan a una estación base (21).

Las arquitecturas presentadas hasta ahora tienen un problema común: los nodos que redireccionan los mensajes lo hacen con un algoritmo de encaminamiento común que tiene que estar disponible y configurado en cada uno de los nodos. Si queremos utilizar una misma red DTN para dos aplicaciones diferentes, o para una única aplicación con más de un tipo de mensaje con encaminamiento diferenciado, encontraremos que la red no se estará utilizando de la mejor manera posible para todos, y que incluso algunos mensajes podrían tener problemas para llegar a su destino. Esto también se aplica para diferentes escenarios. Una red DTN en la que los nodos se van encontrando con cierta periodicidad requerirá un protocolo de encaminamiento diferente de aquella en la que los nodos no pueden predecir cuáles serán los encuentros futuros.

Históricamente, esto ya ha sucedido en otras ocasiones. Por ejemplo, para las redes inalámbricas *ad hoc* tenemos un gran número de protocolos de encaminamiento disponibles (AODV, AORP, ZRP, DSR, etc.) que funcionan bien, cada uno para determinado tipo de aplicaciones o de escenarios. También sucede en el caso del esquema de encaminamiento multidestino (*multicast*) de IP. Aquí volvemos a tener varios protocolos (DVMRP, MOSPF, PIM-DM, PIM-SM, etc.) incompatibles entre sí que irán bien para escenarios o aplicaciones concretas, pero ninguno puede aplicarse siempre. En ambos casos, *ad hoc* y multidestino, el uso simultáneo de estos protocolos no es un caso previsto, y siempre hay que elegir uno, que favorecerá ciertos escenarios o aplicaciones y desfavorecerá otros.

Si queremos una red DTN de propósito general, deberíamos disponer de una variedad de algoritmos de encaminamiento que coexistiesen, y que los mensajes los utilizasen según sus necesidades. Algunas arquitecturas DTN se aproximan a este concepto con unos protocolos prefijados en los nodos que los mensajes pueden elegir. El problema es que añadir nuevos protocolos en una red DTN con nodos no conectados permanentemente quizá no sea posible o requiera mucho tiempo. Por otro lado, en cualquier momento pueden aparecer aplicaciones nuevas con necesidades de encaminamiento particulares. Una alternativa son las redes DTN basadas en mensajes activos, que utilizarán el código móvil para resolver tales problemas.

4.3.1. Mensajes activos

En las redes DTN basadas en mensajes activos, el código de encaminamiento que tendrá que decidir cuál es el mejor nodo vecino para que un mensaje continúe su camino hacia cierto destino viajará junto con el propio mensaje. Es decir, cada mensaje tendrá una parte de información, que es la que deberá llegar al destino, y una parte de código, que se ejecutará en cada nodo y que decidirá la mejor opción entre una serie de nodos vecinos (figura 13).

Figura 13. Nodos en arquitecturas DTN basadas en mensajes activos

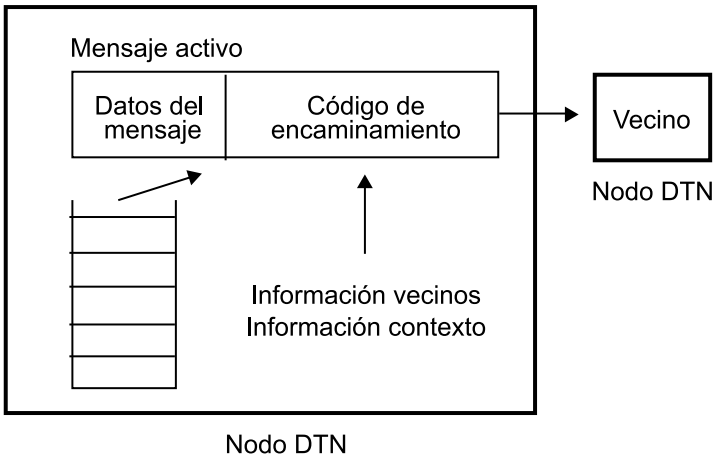


Figura 13

En las redes basadas en mensajes activos, la decisión sobre cuál es el mejor nodo vecino para redireccionarse la toma el propio mensaje mediante su código de encaminamiento.

La aplicación, por tanto, podrá incidir en la forma de utilizar la red DTN poniendo un algoritmo determinado en los mensajes que utiliza. De este modo es posible tener redes DTN de propósito general, en las que ninguna aplicación está obligada a seguir un protocolo común.

El cambio conceptual de estas arquitecturas con respecto a otras más clásicas es profundo. Hasta ahora, el encaminamiento era una función de la capa de red del modelo OSI que utilizaba información disponible solo en esta misma capa. Con las arquitecturas de DTN basadas en mensajes activos, la información utilizada puede provenir de la misma capa y de la capa de aplicación, donde es más probable saber mejor cómo está funcionando la red DTN y qué estrategia de encaminamiento será más acertada. Además, también se encuentra disponible la información sobre el contexto proporcionada por el mismo nodo, que puede ser decisiva a la hora de determinar el siguiente nodo en algunas aplicaciones.

En la figura 13 se puede observar el esquema general de funcionamiento de un nodo DTN utilizando esta arquitectura. Los mensajes pendientes de ser enviados están en una cola de salida. Cuando hay vecinos al alcance, el nodo DTN elige el mensaje que hay que enviar, ejecuta su código de encaminamiento y le pasa como argumento la lista de vecinos y otra información contextual. La función de decisión, en el código del mensaje, devolverá la lista de nodos a

los que interesa saltar. Normalmente, esta lista estará compuesta por solo un elemento, pero el algoritmo de encaminamiento particular podría elegir no ir a ninguno de los vecinos disponibles, o saltar a más de uno (clonándose). Entonces, el nodo DTN intentará enviar el mensaje al nodo seleccionado y pasará a procesar el siguiente.

Para construir este tipo de redes DTN, es preciso que los nodos dispongan de un entorno de ejecución de código. Para los casos en los que los mensajes no lleven código, el nodo puede tener una estrategia de encaminamiento por defecto. Ciertos *frameworks* de desarrollo facilitan en gran medida la creación de nodos DTN con soporte de mensajes activos. Nos referimos a las plataformas de ejecución de agentes móviles, como JADE y Mobile C.

4.3.2. Un ejemplo práctico: PROSES

PROSES (*Protocols for the Single European Sky*, 2010-2012) fue un proyecto financiado por el Ministerio de Industria, Comercio y Turismo del Gobierno de España, y en el que participaron socios académicos y de la industria. El objetivo del proyecto era la creación de una red DTN basada en mensajes activos de la que pudiesen formar parte aviación comercial, aviación general y vehículos aéreos no tripulados.

Aunque pueda parecer que los aviones comerciales están conectados constantemente a través de algún enlace satelital, la realidad nos muestra que esto no siempre es posible, y que cuando lo es, su uso es muy limitado. El enlace satelital tiene un coste muy elevado, y se utiliza principalmente para funciones de gestión aérea. Por otro lado, este enlace no siempre es posible, como cuando las aeronaves vuelan lejos del ecuador. A partir de ciertas latitudes, y en los vuelos que atraviesan las zonas polares, no hay cobertura, y se deja de tener posibilidad de transmisión de datos digitales (entonces se utiliza la radio de voz convencional, con grandes limitaciones). El equipamiento necesario para realizar estas comunicaciones también resulta caro, y por lo general es inaccesible para vehículos pequeños en aviación general.

El proyecto PROSES plantea crear una red DTN de propósito general formada por los vehículos que ocupan el espacio aéreo y basada en mensajes activos. El software creado está basado en el entorno de desarrollo de agentes JADE, y utiliza comunicaciones inalámbricas IEEE 802.11 para las transmisiones entre vehículos. En la parte experimental se han utilizado vehículos aéreos no tripulados (figura 14), donde un equipo dedicado a bordo actúa como nodo de la red, y es independiente del equipo de control y piloto automático. Aparte del pequeño ordenador, cada nodo tiene un dispositivo inalámbrico de conexión a la red, una antena y un receptor GPS.

Figura 14. Vehículos aéreos no tripulados (UAV) en PROSES

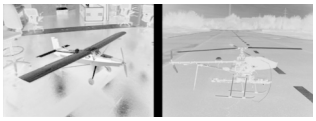


Figura 14

Vehículos aéreos no tripulados (UAV) utilizados en los vuelos experimentales del proyecto PROSES, fotografiados en las instalaciones de uno de los socios del proyecto. Estos vehículos formaban una red DTN aérea, en la que tenían conectividad entre ellos solo si volaban cerca.

Cada nodo va anunciando su presencia periódicamente, por medio de mensajes baliza (*beacon*). En este tipo de mensajes se envía información sobre el nodo mismo, información que se podrá emplear para el encaminamiento. El centro de coordinación del proyecto para los experimentos estaba situado dentro de una furgoneta preparada para contener los equipamientos (figura 15), y actuaba como un nodo más de la red.

En el marco del proyecto, se han desarrollado algunos protocolos y aplicaciones para analizar el funcionamiento de la red y compararlo con los resultados previos obtenidos en las simulaciones.

Figura 15. Estación de control y detalle del equipamiento

Figura 15

A la izquierda, la base de control de los experimentos de PROSES, que estaba dentro de una furgoneta preparada especialmente para ello. Esta base era un nodo más de la red DTN. A la derecha, detalle del ordenador montado sobre uno de los UAV.

Una aplicación consistía en enviar mensajes originados asíncronamente en los UAV y que tenían que llegar a tierra lo antes posible. En este caso, el protocolo de encaminamiento que llevaban los mensajes elegía los nodos con un tiempo previsto de llegada a destino menor.

Otra aplicación enviaba un mensaje a todos los UAV que cumpliesen ciertas condiciones. El protocolo de encaminamiento, a diferencia del caso anterior, era epidémico, y los mensajes iban a todos los vecinos disponibles.

4.4. Investigaciones en curso y problemas abiertos

Todavía hay mucha investigación en marcha en el campo de los protocolos DTN. Aparte del desarrollo de las aplicaciones DTN, hay dos problemas abiertos en dichos protocolos que revisten gran importancia: el encaminamiento y la seguridad.

Por lo que respecta al encaminamiento, ninguno de los protocolos especificados, ni el BP ni el LTP, detalla cómo se pueden establecer las rutas entre los nodos que se comunican. El tratamiento de la conectividad intermitente de las DTN requiere un cambio de paradigma, pasando del modelo clásico de encaminamiento *store-and-forward* a un nuevo modelo *store-carry-forward*. En este nuevo modelo, cuando un paquete de datos llega a un nodo, el salto siguiente para dicho paquete puede no estar disponible inmediatamente, por lo que el nodo deberá almacenarlo y cargar con él, posiblemente durante un periodo de tiempo considerable. La dificultad en el diseño de protocolos que entreguen los paquetes de forma eficiente y satisfactoria a sus destinos estriba en determinar, para cada paquete, el mejor nodo y el mejor momento para reenviar, aspectos no evidentes cuando los nodos no tienen conexión continua con otros nodos. Toda esta nueva funcionalidad en el encaminamiento se sitúa en la capa de aplicación o en la capa superpuesta en la arquitectura del DTNRG. Actualmente, este problema se mantiene muy activo, con enfoques muy diversos, según se consideren redes de dispositivos fijos o de dispositivos móviles, y tanto si son con evolución determinista (encaminamiento prefijado) o con evolución estocástica (encaminamiento probabilístico). Para dispositivos fijos, una posibilidad es usar el modelo de mulas de datos de las redes clásicas de sensores, es decir, elementos móviles que recogen los datos de estos sensores y los conducen a su destino. La forma de combinar esta elección de protocolo de encaminamiento de manera dinámica, adaptándose al tipo de mensaje y a las características específicas de la misma red, parecen conducir a las arquitecturas DTN basadas en mensajes activos.

Estas arquitecturas que utilizan código en los mensajes para llevar a cabo el autoencaminamiento[17]presentan una serie de ventajas claras, como la creación de redes DTN de propósito general o la coexistencia de protocolos de encaminamiento, pero todavía no solucionan totalmente los problemas. Veamos algunas líneas de investigación en este sentido.

[17]Consultad el subapartado 4.3.

Para funcionar lo mejor posible, muchos protocolos de encaminamiento tienen que intercambiar información entre nodos. Esta información se da en toda la red, y es entendida por todos los nodos. Ahora bien, si se utilizan mensajes activos, la información empleada para el encaminamiento pertenece directamente a la cabeza de aplicación. Esto tiene dos implicaciones directas: la primera, que la cantidad de información ya no depende del número de nodos de la red, sino del número de aplicaciones, que puede ser sensiblemente superior, y la segunda, que los nodos no entenderán esta información, y por tanto no podrán operar con ella (fusión de información, descartar redundancias, adaptar parámetros propios, etc.). Dado que gran parte de esta información de encaminamiento deberá estar simultáneamente en el nodo, se utilizan ontologías para su organización y gestión. La forma en que los nodos se intercambian informaciones ontológicas es un problema similar al encaminamiento

Ved también

Dedicamos el apartado 5 a presentar las cuestiones relacionadas con la seguridad en los protocolos tolerantes a retardos e interrupciones.

de mensajes. En este caso, se podría llegar a la conclusión de que la ontología también puede incluir código para especificar cómo se intercambiarán estos tipos de mensajes entre nodos, tal como sucede en el encaminamiento.

5. Problemas de seguridad en protocolos DTN

Los efectos del enfoque DTN sobre la seguridad son inmediatos. Así, cualquier infraestructura centralizada para la gestión de las claves criptográficas pierde su sentido, por la imposibilidad de interactuar de forma continua con el centro de la infraestructura. De manera relacionada, cualquier protocolo clásico basado en una negociación de varios pasos deja de poder utilizarse en entornos DTN.

Lectura recomendada

Como el protocolo TLS, definido en una RFC:

Dierks, T.; Rescorla, E. (2008). "The transport layer security (TLS) protocol version 1.2". *RFC 5246*.

Sin duda, la seguridad se convierte en uno de los problemas abiertos más importantes por resolver en este enfoque (3; 16). Además, dado que cada entorno de conectividad intermitente tiene sus propias especificidades, las soluciones de seguridad deberán ser también concretas para cada entorno considerado. Así, no es lo mismo considerar la seguridad de las comunicaciones interestelares que la de una aplicación de correo en una zona rural sin infraestructura de comunicación.

Las únicas soluciones de seguridad generales publicadas hasta ahora son los mecanismos internos en los diferentes protocolos DTN estandarizados, BP y LTP, para asegurar la privacidad y autenticidad de los datos enviados. Estos mecanismos son similares a los utilizados por IPSec, si bien con claves compartidas previas de larga duración, y no tratan el problema inicial de la gestión de claves.

Nota

Los estándares de seguridad para los protocolos LTP y BP son los RFC 5327 y 6257, respectivamente.

A continuación describiremos las soluciones parciales planteadas para los principales problemas de seguridad de los entornos DTN: por un lado, la gestión de las claves criptográficas y por otro, la seguridad relativa al nuevo modelo de encaminamiento DTN.

5.1. Gestión de claves

Los mecanismos tradicionales basados en infraestructuras de clave pública (*public key infraestructures*, PKI) claramente no son válidos porque obligan a cada participante a contactar con un centro de confianza varias veces, al principio para obtener la firma de una autoridad de certificación sobre su identidad y su clave pública (certificado digital), y después para obtener las claves públicas de otros participantes y/o verificar la validez de los certificados que son presentados por otros participantes. En un entorno DTN no se puede garantizar ninguna cota temporal para estos contactos con el centro de confianza.

Dada esta situación, los únicos esquemas de gestión de claves inmediatamente aplicables son equivalentes a esquemas de compartición de secretos o bien esquemas de clave pública irrevocable, simples pero poco seguros. Algunas pro-

(18)En inglés, *identity based cryptography* (IBC).

puestas de solución utilizan adaptaciones de esquemas basados en criptografía basada en la identidad[18] que solucionan parcialmente el problema. Otras propuestas incluyen el uso de esquemas descentralizados, como SPKI/SDSI[19].

(19)Del inglés, *simple public key infrastructure / simple distributed security infrastructure*.

5.1.1. Soluciones basadas en IBC

En una infraestructura basada en IBC, la clave pública de un usuario se obtiene de manera directa a partir de la identidead de dicho usuario, que puede ser una simple cadena de texto. En estas arquitecturas, las claves privadas son generadas por una tercera entidad de confianza, llamada generador de claves privadas[20]. Al principio, el PKG difunde una clave pública maestra y guarda la clave privada maestra correspondiente. La clave pública de cualquier usuario puede ser fácilmente calculada mediante la combinación de la clave pública maestra y la identidad de dicho usuario. La clave privada correspondiente debe ser solicitada por el usuario al PKG, que la calcula combinando la clave privada maestra con su identidad (ved la figura 16).

(20)En inglés, *private key generator* (PKG).

Figura 16. Criptografía basada en la identidad

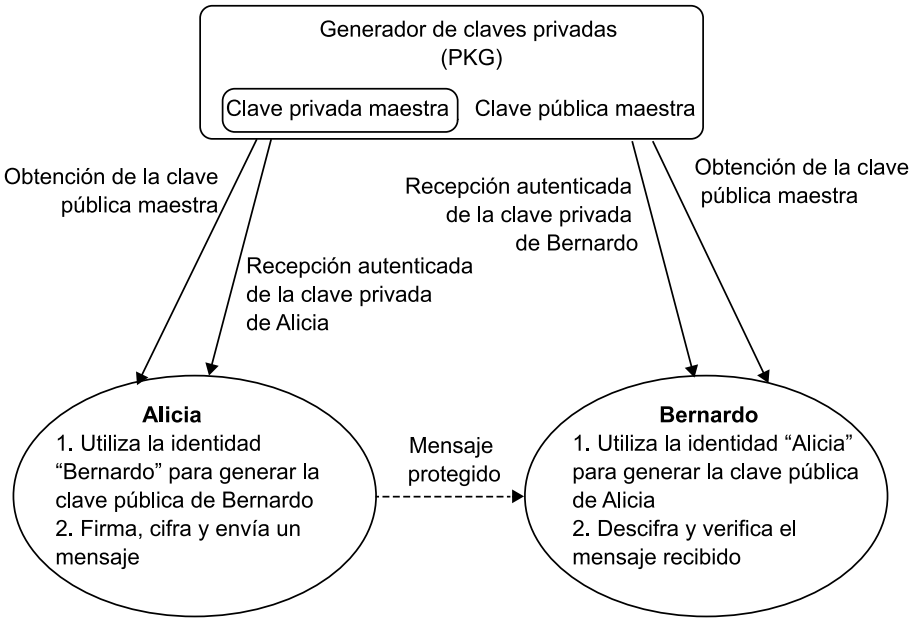


Figura 16

Relación entre los diferentes participantes en una comunicación segura por criptografía basada en la identidad. Alicia y Bernardo pueden enviarse directamente mensajes cifrados y firmados con una interacción inicial, que puede ser fuera de línea, con el PKG.

También hay que destacar que el PKG tiene la capacidad de generar la clave privada de cualquier usuario y que, por tanto, este módulo tiene que ser de total confianza.

Vemos que no hace falta una distribución de claves públicas entre los usuarios para que estos puedan cifrar mensajes y verificar firmas, y por tanto la IBC es adecuada a priori para entornos DTN, siempre que se encuentre una solución al problema de la distribución previa de la clave pública maestra y del acceso de cada usuario al PKG para obtener su clave privada.

Para resolver esta distribución previa, algunas propuestas incluyen la utilización de HIBC[21]. HIBC ofrece una arquitectura distribuida, con una jerarquía de PKG. De este modo, incluso en el caso de que no todos los nodos de la red estén interconectados, todavía es posible adquirir una clave de algunos de los PKG distribuidos.

(21)Del inglés, *hierarchical identity based cryptography*.

En HIBC, se utiliza un PKG raíz y se crean varias subregiones, cada una con su propio PKG local. El PKG raíz establece inicialmente una serie de parámetros para toda la red. A partir de ese momento, el PKG raíz solo necesita generar claves privadas para los PKG de nivel inferior, y estos PKG de subregión generan claves privadas para los usuarios. Los nodos que actúan como usuario son añadidos a una región y esto se convierte en parte de su identidad. Además, para tener en cuenta la posibilidad de que haya nodos en una región completamente desconectada, sin acceso directo a su PKG local, otras propuestas introducen una nueva entidad física, el operador de quiosco, responsable de la validación de la identidad de los usuarios y de la distribución de dispositivos USB que contengan un conjunto de credenciales. Dichas credenciales se utilizan después para obtener las claves IBC utilizando como medio la propia red DTN.

Otras aproximaciones deciden prescindir completamente del servicio centralizado PKG. En estos planteamientos se defiende que el establecimiento de relaciones de confianza se puede hacer utilizando material obtenido de una capa de red inferior, en los momentos en que dicha red esté disponible. Un ejemplo sería un identificador de tarjeta SIM (en redes de teléfonos móviles).

Estos últimos planteamientos asumen un modelo de comunicación organizado en diferentes fases, como mínimo una fase inicial corta de configuración (*setup* o *bootstraping*) con conectividad a una infraestructura de base, y una fase larga de funcionamiento desconectado, donde prevalece el enfoque DTN.

5.1.2. Soluciones basadas en SPKI/SDSI

El último tipo de enfoques aplicados consiste en el uso de esquemas descentralizados como SPKI/SDSI. En estas arquitecturas, la gestión de claves se realiza para todos los miembros de la red de forma distribuida. Este tipo de aproximaciones se ha utilizado, por ejemplo, en el proyecto Haggle, anteriormente descrito. La solución dada a la distribución de claves se basa en certificados de atributos que se utilizan tanto en la autenticación, para validar nodos y de-

mostrar que el titular dispone de los atributos especificados, como para confidencialidad, ya que se utilizan para almacenar la clave pública que se utilizará para establecer conexiones seguras.

Para la distribución de certificados se utiliza una serie de nodos repartidos por la red que actúan como emisores. Todos los nodos de la red conocen a priori su clave pública y pueden obtener certificados de manera segura cuando entran en contacto con alguno de estos nodos de distribución.

5.1.3. Revocación de claves

La revocación de claves plantea también problemas porque la distribución de las tradicionales listas de revocación[22] no es fácil en entornos DTN. La tendencia mayoritaria prevé la utilización de claves criptográficas de corta duración (incluso de pocas horas, en función de la aplicación) para hacer innecesarias las CRL. En el caso de utilizar criptografía basada en la identidad, las claves pasan a depender no solo del identificador de cada usuario, sino también de una marca de tiempo. Las claves comprometidas no se revocan, simplemente caducan al poco tiempo y no se renuevan.

(22)En inglés, *certificate revocation list* (CRL).

5.2. Encaminamiento

El algoritmo de encaminamiento en una red DTN será determinante en las propiedades de seguridad del sistema. Algunas estrategias de encaminamiento pueden ser diseñadas sin mecanismos de protección de la confidencialidad ni de autenticación (5), pero otras pueden necesitarlos para su correcto funcionamiento. La utilización o no de autenticación y protección de la confidencialidad en el proceso de encaminamiento no quita que sean necesarias también de extremo a extremo, es decir, entre aplicaciones.

Tal como sugiere Farrell (3), es preciso que los protocolos y las implementaciones soporten mecanismos de encaminamiento basados en políticas. Es decir, cada protocolo DTN debería especificar qué variables de seguridad se tienen que considerar desde el punto de vista del encaminamiento, de manera que las implementaciones puedan tomar decisiones sobre el envío y reenvío de los mensajes. Este hecho reviste especial relevancia si consideramos que el reenvío o almacenamiento de un mensaje supone un gasto de recursos. En el caso DTN, el reenvío de mensajes tendrá siempre un uso de recursos asociado, como el espacio de memoria o disco necesario, la energía consumida, etc., por lo que se debería incoporar una política de encaminamiento que permitiese tomar decisiones considerando tales variables. Por ejemplo, un nodo intermedio podría exigir que todos los mensajes de entrada estuviesen autenticados, y los mensajes que no cumpliesen el requisito serían rechazados.

Hasta ahora hemos visto que, a pesar de disponer de varios mecanismos para el transporte de la información y su encaminamiento básico en redes DTN, todavía no es posible aprovechar al máximo sus posibilidades. El motivo es

doble: por un lado, necesitamos que los mensajes se dirijan hacia los nodos que ofrecen una probabilidad más alta de que el mensaje llegue a su destino con las restricciones impuestas por la misma red y por la aplicación; por otro lado, son necesarios esquemas de seguridad que sean tolerantes con la no contemporaneidad de las conexiones entre nodos, característica en DTN. Por los dos hechos expuestos, todavía no existe un estándar consensuado, pero hay bastantes propuestas para obtenerlo. En apartados anteriores hemos visto algunas alternativas para efectuar el encaminamiento, entre las cuales se destacan las arquitecturas basadas en mensajes activos, que ofrecen una importante flexibilidad para este tipo de redes.

Por lo que respecta a la seguridad, se pueden aplicar las técnicas habituales de cifrado y autenticación, teniendo en cuenta las consideraciones particulares en este tipo de redes ya vistas al principio del presente apartado. Se plantea otra cuestión muy importante en las redes DTN: la responsabilidad que adquiere un nodo cuando acepta transportar un mensaje, es decir, ser su custodio. Esto es nuevo en el modelo *store-carry-and-forward*, ya que está asociado al concepto de *carry*. Desde el punto de vista de la seguridad, podemos considerar esquemas de no repudio que obtengan recibos criptográficos que permitan reconstruir trazas de mensajes a posteriori, por ejemplo, y detectar los nodos que no hayan cumplido sus compromisos y actuar contra ellos.

5.2.1. No repudio

La primera dificultad que encontramos a la hora de aplicar un esquema de no repudio en DTN es que no podemos basarnos en ninguna propuesta que utilice una tercera parte de confianza por la naturaleza misma de este tipo de red. Por otro lado, en un protocolo de transmisión no repudiable de mensajes, hay que garantizar que tanto el mensaje como el justificante de su recepción llegan a las partes correspondientes de forma simultánea. Así pues, tenemos que conseguir un protocolo que asegure esta concurrencia y que no utilice una tercera parte de confianza.

En criptografía, el problema del intercambio justo de firmas, en el que las firmas de emisor y receptor quedan vinculadas al mismo tiempo, ya ha sido estudiado previamente. En este tipo de esquemas, ni emisor ni receptor pueden retractarse de la comunicación, y de esta forma proporcionan no repudio.

Susilo y otros (22) introducen el concepto de las firmas concurrentes perfectas, en las que el emisor y el receptor pueden producir firmas ambiguas que no son vinculantes hasta que uno de los dos libera una pieza extra de información. Directamente, estos esquemas no se pueden utilizar para el intercambio de mensajes en DTN por la necesidad de acceso a una tercera parte de confianza, pero admiten variaciones utilizando criptografía basada en identidad, por ejemplo[23], donde las claves públicas son un rasgo identificativo de los nodos

[23] Ved el subapartado 5.1.1.

y, por tanto, conocido. Chow y Susilo (9), por ejemplo, proponen un esquema eficiente para el intercambio justo de claves con firmas concurrentes perfectas utilizando IBC.

Para adaptar estos esquemas al problema del no repudio en el encaminamiento en redes DTN, se tienen que efectuar algunas modificaciones. Martínez-Bea y otros (15) proponen una variante que utiliza el mismo mensaje para ser enviado como pieza final de información que permite validar las firmas del esquema, y por tanto los justificantes de recepción/envío. Veamos con más detalle cómo funcionaría el esquema. Si el nodo Alice quiere enviar un mensaje al nodo Bob para que lo retransmita, antes que nada necesitamos:

1) Elegir dos número primos grandes p y q tales que $q|p-1$.

2) Elegir un g de orden q tal que $g \in \mathbb{Z}_p^*$.

3) Tanto para Alice (A) como para Bob (B), generar el par de claves (sk_i, pk_i) tal que $pk_i = g^{-sk_i} mod p$, donde sk_i es la clave privada y pk_i, la clave pública. Así, Alice, por ejemplo, tendrá el par *skA* (pública) y *pkA* (privada).

4) Tanto Alice como Bob deben tener preparadas sus claves IBC, *ipk*, clave pública, e *isk*, clave privada.

Hecho esto, tanto Alice como Bob pueden preparar sus respectivas firmas y enviarlas:

1) $k = H(mensaje|Sal)$, donde $H()$ es una función resumen y Sal será un número aleatorio.

2) Escoger un w tal que $w \in \mathbb{Z}_p^*$.

3) Calcular $m = \langle E_{ipk}(pk), S_{isk}(H(pk)) \rangle$, donde *ipk* es la clave pública IBC del receptor, *isk* la privada IBC del emisor, *pk* la clave pública del emisor y $E()$y $S()$, funciones de cifrar y firmar, respectivamente.

4) Calcular $r = g^w mod p$.

5) Calcular $e = H(m, k, r)$, donde H es una función resumen.

6) Calcular $c = w + esk mod q$, donde *sk* es la clave privada de firma.

7) Construir firma $\sigma = \langle m, k, s \rangle$, donde $s = \langle r, e, c \rangle$.

Finalmente, una vez intercambiados σ_A y σ_B, Alice envía el mensaje y la sal, de modo que ya quedará todo preparado para verificar las firmas. Alice habrá conseguido enviar el mensaje a Bob, y los dos tendrán un justificante que podrá demostrar ante un tercero la acción que llevaron a cabo (enviar/recibir). La figura 17 muestra el funcionamiento del protocolo en sus tres pasos.

Figura 17. Protocolo criptográfico para el encaminamiento no repudiable

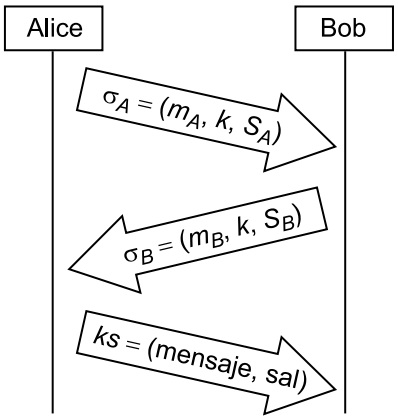


Figura 17

Mediante un protocolo de intercambio simultáneo de firmas, es posible obtener justificantes, que impedirán que ninguna de las partes se retracte de la comunicación.

5.3. Control de acceso en los nodos DTN

En las arquitecturas DTN basadas en mensajes activos, en los que el código de encaminamiento acompaña al propio mensaje, es preciso acceder a información almacenada en el nodo para poder tomar la decisión de encaminamiento. Normalmente, esta información estará estructurada en ontologías, que facilitarán al nodo las tareas para su gestión, actualización y utilización.

Ved también

Las arquitecturas DTN basadas en mensajes activos se estudian en el subapartado 4.3.

En este punto, se presenta un problema de seguridad importante. La información que se tiene que utilizar desde aplicaciones concretas puede ser el objetivo de un atacante que pretenda alterar el funcionamiento de una aplicación particular. Así pues, se hace preciso controlar el acceso a esta información para evitar que pueda ser consultada o modificada desde un mensaje no autorizado. Este control de acceso puede estar basado en IBC para garantizar la confidencialidad y la integridad de la información utilizada durante el encaminamiento, o se puede utilizar un esquema más sencillo basado en el mismo código de encaminamiento. Sería como un control de acceso biométrico por ADN, si consideramos el código de encaminamiento como el ADN del mensaje.

Ved también

La IBC se estudia en el subapartado 5.1.1 de este módulo.

Dos mensajes diferentes que comparten el código de encaminamiento tendrán acceso a la misma información. Esto debe ser así porque los dos códigos tienen que poder acceder a la misma información de encaminamiento.

Encontramos un ejemplo de control de acceso con estas características en la obra de Sánchez-Carmona y otro (20). En esta propuesta, la información está cifrada en el nodo, y se dispone de una lista de reglas para el control de acceso. Cuando un mensaje solicita cierta información durante el encaminamiento,

se aplicarán dos funciones de resumen criptográfico diferentes sobre el código que ha solicitado la información. Estos valores se usarán para recuperar la clave simétrica que permite descifrar la información.

Veamos cómo funciona con más detalle. Las informaciones de encaminamiento de cierto protocolo estarán cifradas e indexadas del modo siguiente:

$$(j, E_{k_j}(I_j)) \qquad 6.10$$

Donde el primer elemento de cada dupla sirve para identificar las diferentes I_j y poder realizar una busca rápida de una información concreta, y el segundo elemento, I_j, es la información de encaminamiento en cuestión, debidamente cifrada para que ningún código c_i (u otro proceso) sin autorización pueda acceder a ella.

Las reglas de control de acceso son, así:

$$(j, h'(c_i), E_{h(c_i)}(k_j)) \qquad 6.11$$

Donde:

- j es el identificador de la información I_j.
- E es un algoritmo de cifrado con clave simétrica.
- h y h' son dos algoritmos de resumen diferentes.
- c_i hace referencia al código de encaminamiento del mensaje i.
- k_j es la clave simétrica necesaria para cifrar y descifrar la información I_j sobre la que se controla el acceso.

Ahora solo hay que definir cómo puede acceder el código c_i a la información I_j en este esquema:

1) Se calcula $h'(c_i)$ y se busca la regla de control de acceso correspondiente. En este caso, podría ser $(j, h'(c_i), E_{h(c_i)}(k_j))$.

2) Se calcula $h(c_i)$ para descifrar la clave de acceso k_j.

3) Con k_j ya se puede descifrar $E_{k_j}(I_j)$ para obtener I_j.

Este esquema permite que la información de encaminamiento sea accesible solo para cierto código, y que incluso en caso de que un nodo resultase comprometido, no sería posible acceder a esta información (necesitaremos un mensaje con el código autorizado). Para confeccionar un código malicioso que intenta-

se envenenar la información de encaminamiento haciéndose pasar por código lícito sería necesario llevar a cabo un doble ataque de preimagen simultáneo en las dos funciones resumen (uno para engañar al sistema de indexación de claves y el otro, para el descifrado de la información). La probabilidad de tener éxito en un doble ataque de estas características, teniendo en cuenta que son funciones resumen diferentes, es muy baja.

Resumen

En este módulo didáctico hemos hecho referencia a mecanismos y protocolos de seguridad para redes formadas por dispositivos inalámbricos con limitaciones por lo que respecta a autonomía y capacidad de cálculo, potencialmente móviles, o sin conectividad permanente entre las aplicaciones en comunicación.

Por lo que respecta a las redes *ad hoc*, hemos tratado el problema de establecer una red con nodos desconocidos. En primer lugar, hemos visto cómo funcionan los **mecanismos de descubrimiento seguro de nodos vecinos**. A continuación hemos analizado los **protocolos de encaminamiento** y hemos descrito cómo funcionan las extensiones de seguridad de estos protocolos. Finalmente, hemos explicado cuáles son los riesgos que comportan estas redes para la **privacidad** de los usuarios y cuáles son los mecanismos que se pueden implementar para proteger su identidad y localización.

En lo concerniente a las redes de sensores, hemos analizado diferentes **protocolos de gestión de claves criptográficas** para poder establecer las bases de una infraestructura de seguridad en las redes de sensores. También hemos presentado soluciones eficientes basadas en **agregación de datos** para autenticar la gran cantidad de datos que los sensores van acumulando de sus lecturas y que tienen que enviar a una estación base.

Por lo que respecta a los protocolos tolerantes a retardos e interrupciones, hemos visto las diferentes arquitecturas propuestas para tratar las diversas situaciones de conectividad intermitente que se pueden dar, así como las soluciones propuestas para afrontar los problemas de seguridad aún abiertos que plantean dichos protocolos: básicamente, la **gestión de las claves criptográficas** y el **encaminamiento con custodia de la información** en los nodos intermedios.

Actividades

1. Buscad información sobre los mecanismos de Watchdog y Pathrater. ¿Cómo pueden mejorar la seguridad de las redes ad hoc?

2. El protocolo de encaminamiento DSR es vulnerable a ataques de *blackhole*. Explicad por qué el protocolo es vulnerable a dichos ataques, si las extensiones de seguridad Ariadne-MAC y Ariadne-TESLA pueden prevenirlos y, en caso afirmativo, cómo pueden hacerlo.

3. Diseñad una red de tipo DTN para recoger datos de una serie de estaciones meteorológicas dispersas en un entorno rural. ¿Cuáles serían los nodos? ¿Qué estrategia de encaminamiento utilizaríais?

Ejercicios de autoevaluación

1. En una red *ad hoc* que utiliza encaminamiento SAODV con funciones resumen sha-1, dos usuarios reciben los siguientes mensajes RREQ. ¿Son correctos?

a) HopCount=4,
MaxHopCount=255,
Hash (en base64)="EQM3MrVYrwvM+hZoJanqKsrHDD4=",
TopHash (en base64) = "sXLlKsKjqrm5RDVDS14HeF+VXyE="

b) HopCount=5,
MaxHopCount=255,
Hash (en base64)="tzvBaHrNpAph0tKworJdoQ8nbHQ=",
TopHash (en base64)="sXLlKsKjqrm5RDVDS14HeF+VXyE="

2. Indicad qué clave utilizaríais y qué operación efectuaríais para enviar los siguientes mensajes autenticados en una red de sensores:

a) Un nodo envía un mensaje a la estación base.

b) La estación base envía un mensaje en difusión (*broadcast*) a todos los nodos de la red.

c) Un nodo envía un mensaje en difusión (*broadcast*) a todos sus vecinos.

3. ¿Cuál de las siguientes afirmaciones sobre las redes DTN es **cierta**?

a) Para el encaminamiento de mensajes podemos utilizar un protocolo pensado para redes *ad hoc*, porque también son redes emergentes.
b) Se pueden utilizar esquemas de seguridad clásicos basados en PKI para diseñar mecanismos de encaminamiento seguro.
c) La estrategia de encaminamiento debe ser la misma para todos los mensajes en la misma red.
d) En un futuro, las DTN pueden llegar a desplazar los protocolos actuales de Internet.
e) Ninguna de las respuestas anteriores.

Solucionario

Ejercicios de autoevaluación

1. a) Correcto

b) Incorrecto

2. El nodo utiliza la clave de nodo que comparte con la estación base para enviarle un mensaje y un MAC generado con esta clave. Una red de sensores tiene una clave de red que comparten la estación base y todos los nodos. Sin embargo, esta clave no se puede utilizar para autenticar mensajes, ya que un nodo que recibiese un mensaje con un MAC calculado a partir de esta clave no podría estar seguro de quién lo ha generado; cualquier nodo que tenga la clave podría haberlo hecho.

La autenticación en difusión basada en claves simétricas se puede hacer con el protocolo TESLA. Este protocolo debe distribuir el elemento raíz de una cadena resumen a través de un mensaje autenticado. En el caso de hacer una difusión global, el elemento raíz de la estación base se puede precargar en cada sensor antes del despliegue de la red. La autenticación de mensajes de difusión locales es similar al punto anterior, se puede utilizar el protocolo TESLA. En este caso, para establecer el elemento raíz de la cadena resumen, se utilizaría un mensaje autenticado unidestino (*unicast*) entre el nodo y todos sus vecinos (ved la respuesta del primer punto).

3. e

Glosario

ataque DoS Ataque de denegación del servicio (del inglés, *denial of service*).

ataque MITM Ataque del hombre a medio camino (del inglès, *man-in-the-middle*).

ataque Sybil Ataque en el que un usuario adopta diferentes identidades para tener más influencia en la red.

delay and disruption tolerant networking Protocolo tolerante a retardos e interrupciones.
sigla **DTN**

DTN Véase **delay and disruption tolerant networking**.

IBC Véase **identity-based cryptography**.

identity-based cryptography Criptografía basada en la identidad.
sigla **IBC**

MANET *ad hoc* Redesmóviles (del inglés, *mobile ad hoc network*).

paquete RREP Mensaje de respuesta del protocolo de descubrimiento de rutas para crear tablas de encaminamiento (del inglés, *route reply*).

paquete RREQ Mensaje de solicitud de descubrimiento de rutas para crear tablas de encaminamiento (del inglés, *route request*).

wireless mesh network Red con topología de malla.
sigla **WMN**

wireless sensor network Red de sensores.
sigla **WSN**

WMN Véase **wireless mesh networksy**.

WSN Véase **wireless sensor network**.

Bibliografía

Bibliografía básica

Çayirci, E.; Rong, C. (2009). *Security in Wireless Ad Hoc and Sensor Networks*. John Wiley & Sons. http://books.google.co.uk/books?id=3EvhTrocBZUC.

Cerf, V.; Burleigh, S.; Hooke, A.; Torgerson, L.; Durst, R.; Scott, K.; Fall, K.; Weiss, H. (2007). «RFC 4838, Delay-Tolerant Networking Architecture». *IRTF DTN Research Group*.

Farrell, S.; Cahill, V. (2006). *Delay- and Disruption-Tolerant Networking*. Norwood, MA, USA: Artech House, Inc. ISBN 1596930632.

Perkins, C. E. (2008). *Ad Hoc Networking*. (1.ª ed.). Addison-Wesley Professional. ISBN 0321579070, 9780321579072.

Bibliografía complementaria

Burgess, J.; Bissias, G. D.; Corner, M. D.; Levine, B. N. (2007). «Surviving Attacks on Disruption-Tolerant Networks Without Authentication». En: «MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing», (págs. 61–70). Nueva York: ACM. ISBN 978-1-59593-684-4. doi: http://doi.acm.org/10.1145/1288107.1288116.

Buttyan, L.; Hubaux, J.-P. (2008). *Security and Cooperation in Wireless Networks: Thwarting Malicious and Selfish Behavior in the Age of Ubiquitous Computing*. Cambridge University Press.

Chaum, D. (1985). «Security without identification: transaction systems to make big brother obsolete». *Commun. ACM*, volumen 28 (n.º 10, págs. 1030–1044). ISSN 0001-0782. doi: http://doi.acm.org/10.1145/4372.4373.

Chaum, D. L. (1981). «Untraceable electronic mail, return addresses, and digital pseudonyms». *Commun. ACM*, volumen 24 (n.º 2, págs. 84–90). ISSN 0001-0782. doi:http://doi.acm.org/10.1145/358549.358563.

Chow, S.; Susilo, W. (2005). «Generic Construction of (Identity-Based) Perfect Concurrent Signatures». *Lecture Notes in Computer Science 3783: Information and Communications Security*, (págs. 194–206).

Clausen, T.; Jacquet, P. (2003). «Optimized Link State Routing Protocol (OLSR)». RFC 3626 (Experimental). http://www.ietf.org/rfc/rfc3626.txt.

Dierks, T.; Rescorla, E. (2008). «RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2». *IETF*.

Eschenauer, L.; Gligor, V. D. (2002). «A key-management scheme for distributed sensor networks». En: «Proceedings of the 9th ACM conference on Computer and communications security», CCS '02, (pàgs. 41–47). Nueva York: ACM. ISBN 1-58113-612-9. doi: 10.1145/586110.586117. http://doi.acm.org/10.1145/586110.586117.

Farrell, S. (2007). «Lakes, Noise and DTN Protocols: SeNDT & DTN Transport». SeNDT Presentation at Nokia. http://down.dsg.cs.tcd.ie/sendt/SeNDT-LTP-T-20070619.ppt.

Johnson, D.; Hu, Y.; Maltz, D. (2007). «The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4». RFC 4728 (Experimental). http://www.ietf.org/rfc/rfc4728.txt.

Martínez-Bea, S.; Castillo-Pérez, S.; Robles, S.; Gozalbo-Baró, M. (2012). «Protocolo de No-repudio para Redes DTN Basado en Intercambio Justo de Firmas». En: «XII Reunión Española sobre Criptología y Seguridad de la Información», Mondragon Unibertsitatea.

Martínez-Vidal, R.; C. de Toro, M.; Martí, R.; Borrell, J. (2012). «Esquema de gestión de claves criptográficas tolerante a retrasos e interrupciones en entornos aeronáuticos». En: «XII Reunión Española sobre Criptología y Seguridad de la Información», Mondragon Unibertsitatea.

Martonosi, M. (2005). «The ZebraNet Wildlife Tracker». Princeton. http://www.princeton.edu/ mrm/zebranet.html.

Perkins, C.; Royer, E.; Das, S. (2003). «Ad hoc On-Demand Distance Vector (AODV) Routing». RFC 3561 (Experimental). http://tools.ietf.org/html/rfc3561.

Perrig, A.; Song, D.; Canetti, R.; Tygar, J. D.; Briscoe, B. (2005). «Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction». RFC 4082 (Informational). http://www.ietf.org/rfc/rfc4082.txt.

Sánchez-Carmona, A.; Borrego, C.; Robles, S.; Andújar, J. (2012). «Control de Acceso para Mensajes Pro-activos en Redes DTN». En: «XII Reunión Española sobre Criptología y Seguridad de la Información», Mondragon Unibertsitatea.

Shukla, S.; Bulusu, N.; Jha, S. (2004). «Cane-toad Monitoring in Kakadu National Park Using Wireless Sensor Networks». Proc. Network Research Workshop, as part of 18th APAN Meetings, Cairns, Australia. http://www.cse.unsw.edu.au/sensar/publications/kakadu.pdf.

Susilo, W.; Mu, Y.; Zhang, F. (2004). «Perfect Concurrent Signature Schemes». *Lecture Notes in Computer Science 3269: Information and Communications Security*, (págs. 14–26).

Zhang, J.; Varadharajan, V. (2010). «Wireless sensor network key management survey and taxonomy».*Journal of Network and Computer Applications*, volumen 33 (n.º 2, págs. 63–75). ISSN 1084-8045. doi: 10.1016/j.jnca.2009.10.001. http://www.sciencedirect.com/science/article/pii/S1084804509001313.