CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS

DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN COMUNICACIONES

"Análisis, evaluación e implementación de protocolos de autenticación en sistemas de cómputo distribuido"

Tesis que presenta

Ing. Gerardo Díaz Aguilar

Para obtener el grado de

Maestro en Ciencias

Directores de Tesis:

Dr. Domingo Lara Rodríguez

Dr. Raúl García Ruiz

México, Ciudad de México Febrero, 2016

Dedicatoria

A mis padres Braulio y Soledad

A mi hermana Alejandra

A Víctor.

Agradecimientos

A mis padres por todo el apoyo incondicional que siempre han brindado para la realización de mi vida y de mis metas.

A mi hermanita Ale por siempre alegrarme y motivarme a seguir adelante y nunca rendirme.

A mis asesores el Dr. Domingo Lara R. y el Dr. Raúl García R. que gracias a sus conocimientos y consejos hicieron posible la realización de esta tesis.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por apoyarme económicamente durante el período de mi maestría.

Al Centro de Investigación y de Estudios Avanzados del I.P.N. (CINVESTAV) por haberme abierto las puertas y permitirme superarme.

A Angel, Felipe, Víctor, Alavid, Rebeca y Tonatiuh por haberme acompañado y apoyado en este difícil camino de la maestría.

A la señora Lidia y al señor Don Pablo, por abrirme las puertas de su hogar y brindarme su apoyo y amistad en esta nueva etapa de mi vida.

A Víctor R., Rubén, Edgar, Eduardo, Luis, Saida, Ana, y a todos mis amigos que me acompañaron en esta aventura llamada CINVESTAV.

Resumen

Actualmente, muchos de los servicios de cómputo a los que se desea acceder se encuentran fuera de nuestro ordenador, por lo que es común que un usuario el cual desea recibir un servicio requiera identificarse ante una entidad remota. Al proceso de identificación y verificación de usuarios se le conoce como autenticación. La autenticación constituye la primera línea de defensa de todo sistema informático, por lo que es de vital importancia contar con mecanismos seguros para evitar la intrusión de usuarios o entidades desconocidas.

En el presente trabajo, se estudian y evalúan algunas de las estrategias utilizadas para la autenticación de entidades en ambientes de cómputo distribuido. Bajo la terminología de autenticación, una entidad es un dispositivo, usuario humano o proceso de cómputo, el cual requiere tener acceso a alguno de los recursos proporcionados por una o varias redes.

En primera instancia, se presentan algunos de los protocolos de autenticación históricos que sentaron las bases de los protocolos empleados en la actualidad. Debido a la necesidad de formalizar la evaluación de los protocolos de autenticación, se estudian dos herramientas que ayudan a la realización de un análisis más sistemático y estricto de los protocolos de autenticación. La primera de ellas es la herramienta de análisis lógico SVO, la cual ayuda a encontrar vulnerabilidades; y la segunda se trata de un conjunto de principios que ayudan a evitar errores durante la etapa de diseño de los protocolos de autenticación.

Con la finalidad de comprender cómo operan los protocolos de autenticación en ambientes reales, se procedió a la implementación y montaje de escenarios de pruebas físicos para dos de los protocolos de autenticación de uso actual estandarizados por la IETF (*Internet Engineering Task Force*), el protocolo Kerberos, y el protocolo RADIUS (*Remote Authentication Dial-In User Service*).

Finalmente, se presenta un estudio basado en las herramientas de análisis aprendidas para tres protocolos de autenticación. En el primero, se le da un seguimiento a los trabajos realizados previamente sobre el protocolo de Otway-Rees y se proponen algunas modificaciones para mejorar su funcionalidad. El segundo y el tercero, se tratan protocolos de autenticación empleados en ambientes inalámbricos como lo son Wi-Fi y LTE, en los cuales se realiza sus respectivos análisis formales propuestos en este trabajo.

Abstract

At present, many of the application services the people want to access are outside of their computers, so that when a user wants to access a service, he needs to communicate with a remote entity and present identity credentials to it. The process of identification and verification of users is named authentication. The authentication builds the first line of defense in every computing system, and then it is very important to count with security mechanisms in order to avoid intrusion of unknown entities.

This thesis aims the study of the strategies the entities can use to achieve authentication on distributed computer systems. Under authentication terminology, an entity is a device, human user or computer process, which wants to have access to some of the services provided by network computers.

First, this work presents some of the historical authentication protocols that constitute the fundamentals of the newest authentication protocols. Due to the necessity of formalizing the evaluation of authentication protocols, a set of tools are studied in order to have a formal evaluation of them. The first tool studied is called SVO Logic. The main purpose of this logic is to find out vulnerabilities inside the authentication protocols. The second tool studied is a set of principles which helps the designers to avoid the common errors during the design step of cryptographic protocols.

For better comprehension of the authentication protocols, two of the standardized authentication protocols currently active were tested using some of the test-benches proposed in this thesis. The first protocols is Kerberos, proposed at the MIT (*Massachusetts Institute of Technology*), and the second is the protocol RADIUS (*Remote Authentication Dial-In User Service*), developed by Livingston Enterprises, Inc.

Finally, this work presents an analysis using the tools learned to evaluate three authentication protocols. In the first protocol (called Otway and Rees Authentication Protocol), some improvements about its functionality are proposed following the previous issues reported about this protocol. On the next two protocols, abstract models and formal analysis were proposed in order to evaluate the security of the protocols used in wireless environments like Wi-Fi and LTE (*Long Term Evolution*).

Índice general

Índice de figuras

Índice de tablas

Objetivos

La presente tesis tiene como objetivo principal el estudio de las estrategias de autenticación empleadas en ambientes de cómputo distribuido.

Los protocolos de autenticación aquí expuestos van desde los primeros trabajos presentados en el área hasta algunos de los protocolos de uso reciente. Para llevar a cabo el análisis de los mismos, se estudiaron dos herramientas de evaluación de protocolos criptográficos. La comprensión de estas herramientas constituye uno de los objetivos específicos de esta tesis.

Por otra parte, otro de los objetivos de este trabajo es la comprensión práctica de cómo funcionan los protocolos de autenticación en ambientes de operación reales. Para cumplir este objetivo, se procedió al estudio de dos estándares de la IETF (*Internet Engineering Task Force*) de uso común en autenticación y se realizó la implementación de aplicaciones cliente/servidor siguiendo la descripción de los estándares analizados.

Un objetivo que resulta del estudio y comprensión de las herramientas de evaluación formal de protocolos, es el de su uso para encontrar deficiencias en algunos protocolos de autentencación de uso reciente. Para esto, se elaboraron análisis formales de autenticación para dos de los estándares actuales como lo son IEEE 802.11 (Wi-Fi), y LTE (*Long Term Evolution*).

Introducción

La autenticación es seguramente el problema más básico de seguridad para los desarrolladores de protocolos. El problema de autenticación es fácil de describir, pero difícil de resolver: Dos entidades (ya sean dispositivos, usuarios humanos, o procesos de cómputo) que se encuentran comunicándose desean identificarse la una con la otra para establecer una relación de confianza.

Es común hoy en día que la mayoría de los servicios a los cuales deseamos acceder se encuentran fuera de nuestro equipo de cómputo, por lo que para acceder a estos servicios necesitamos presentar cierta información para ser autorizados. Gran parte de las comunicaciones actualmente se realizan a través de medios no seguros como lo es la Internet. A medida que ha ido creciendo el uso de sistemas distribuidos en ambientes de redes basados en comunicaciones con TCP/IP (Internet), la necesidad de contar con estrategias seguras para identificar y verificar usuarios ha ido en aumento.

El objetivo de esta tesis es realizar un estudio a profundidad de las estrategias más comunes de autenticación de entidades. El enfoque que se le da a este trabajo es de carácter teórico-práctico, siendo la parte teórica la caracterización y análisis de los protocolos mediante modelos abstractos, y la parte práctica la evaluación con escenarios de pruebas de algunos estándares de autenticación de la IETF (*Internet Engineering Task Force*).

La tesis esta organizada en cuatro capítulos. En el primer capítulo, se introduce el tema de autenticación. Se presentan los conceptos necesarios para el entendimiento del mismo y se explica la importancia que conlleva realizar este proceso. Además, se describen algunas de las primeras propuestas de protocolos de autenticación que debido a su importancia, constituyen la base para muchos de los protocolos empleados en la actualidad. Gracias a estos protocolos, es posible obtener una mejor comprensión de las implicaciones que tiene la identificación de usuarios en un ambiente abierto e inseguro.

En el segundo capítulo, se estudian y exponen algunas de las herramientas para la evaluación de los protocolos de autenticación. Estas herramientas si bien no resuelven completamente el problema, nos proporcionan una idea más detallada de cuáles podrían ser las causas para que un protocolo presente fallos y pueda ser vulnerado a la hora de ponerse

en marcha en un sistema específico. La primera herramienta de evaluación a tratar, es la conocida como lógica SVO, que presenta mediante formulaciones derivadas de la lógica de proposiciones, una forma de expresar formalmente si un protocolo de autenticación cumple con los requerimientos para los cuales fue diseñado. La segunda herramienta que se trata, es una serie de principios expuestos por Martín Abadi y Roger Needham enfocados al diseño de protocolos seguros de autenticación.

En el tercer capítulo, se describen los resultados de algunas pruebas físicas de implementaciones realizadas de dos de los protocolos de autenticación empleados en la actualidad. Estos son el protocolo Kerberos y el protocolos RADIUS (*Remote Authentication Dial-In User Service*). Kerberos es un protocolo desarrollado en el MIT (*Massachusetts Institute of Technology*) y tiene como objetivo la autenticación mutua de usuarios, normalmente entre clientes y un servidores. RADIUS es un protocolo que hace uso del modelo del protocolo de autenticación extensible (EAP por sus siglas en inglés) para implementar diversos métodos de autenticación entre entidades. Para demostrar el funcionamiento y características de RADIUS y Kerberos, se montaron algunos escenarios de pruebas en los cuales se utilizan estos protocolos para la validación de usuarios.

Finalmente, en el capítulo cuatro, se analizan algunos de los protocolos de autenticación empleados en ambientes inalámbricos como lo son Wi-Fi y LTE (*Long Term Evolution*). Este análisis se lleva a cabo mediante el uso de las herramientas de estudio de protocolos de autenticación previamente expuestas. Así mismo se realiza un estudio a profundidad de uno de los protocolos presentado en los primeros trabajos dentro del área, que es el protocolo de Otway y Rees. Se le da un seguimiento a las mejoras realizadas por otros autores en trabajos más recientes y además se proponen algunas mejoras adicionales con la ayuda de las herramientas estudiadas.

Capítulo 1

Principios de autenticación de entidades en sistemas de cómputo distribuido

En el presente capítulo se exponen los principios básicos relacionados al tema de autenticación de entidades. Dentro de este capítulo y posteriores, la palabra entidad se utilizará para referirse ya sea a una máquina, un usuario humano o proceso de cómputo. En primer lugar, se presenta un panorama general del problema de seguridad en redes, de los tipos de ataques a los que pueden ser sometidas así como de los requerimientos necesarios para protegerlas (uno de los cuales es la autenticación). A continuación se hablará acerca de la criptografía. Esta constituye la base de la seguridad en las comunicaciones dentro de los sistemas de cómputo distribuido. Finalmente, se concluye el capítulo explicando algunos conceptos y estrategias de autenticación entre entidades, así como algunas de las propiedades de seguridad de los protocolos de autenticación.

1.1. Seguridad en redes de cómputo

En la actualidad se presentan muchos problemas de seguridad a la hora de establecer y llevar a cabo la comunicación entre dos equipos (o usuarios) que se encuentran en una misma red o en diferentes redes de computadoras. Un adversario (o intruso) que quiera interferir en la comunicación puede efectuar varios tipos de ataques los cuales pueden causar diversos problemas a las partes comunicantes. Dichos ataques pueden ser clasificados en dos grupos principales [1]:

- **Ataques pasivos.** En estos, un adversario intenta obtener y hacer uso de la información que se está transmitiendo sin afectar al sistema o a los mensajes enviados. Dos tipos de ataques pasivos son los de revelación de los mensajes (espionaje) y análisis de

tráfico (análisis de la información). Dada la naturaleza de los ataques pasivos, resulta muy difícil poder detectarlos ya que no involucran ninguna alteración de la información.

- **Ataques activos.** Un ataque activo involucra modificar el flujo de los datos o la creación de un flujo falso y puede ser subdividido en cuatro categorías: enmascaramiento (hacerse pasar por alguien más), repetición, modificación de los mensajes y denegación del servicio. Este tipo de ataques son más fácilmente detectables pero son muy difíciles de prevenir debido a la amplia variedad de posibles vulnerabilidades que existen tanto el la parte de hardware y el software de las distintas entidades.

Cada uno de estos ataques pueden ser llevados acabo dependiendo de cómo se encuentren estructuradas las redes y pueden afectar de manera significativa a las entidades participantes en la comunicación. Un ejemplo muy claro es la internet, que debido a que se encuentra formada por la interconexión de miles de redes, la comunicación entre dos usuarios que se encuentran en diferentes redes debe atravesar por diversas redes las cuales pueden ser o no confiables. Esto puede ocasionar que la información transmitida en el intercambio de mensajes se encuentre comprometida.

Dada esta problemática, existen grupos de trabajo que se han dado a la tarea de implementar estrategias para contrarrestar y prevenir los ataques hacia las redes, tratando este problema bajo el nombre de "seguridad en redes". Este término involucra dos cosas principalmente, una es la protección del sistema contra accesos de entidades no permitidas (como pueden ser los hackers o los virus por la parte de software), y la otra es el establecimiento de comunicaciones seguras.

Enfocándonos en la parte de comunicaciones seguras, supongamos que tenemos dos usuarios que se desean comunicar de forma segura, a los que llamaremos A y B. A desea comunicarse con B de tal manera que sólo B pueda entender el mensaje que A le envía, aunque ellos se comuniquen a través de un medio inseguro en donde un intruso denominado D pueda interceptar cualquier mensaje que sea transmitido de A hacia B (figura 1.1). B también quiere asegurarse que el mensaje que recibe de A fue en realidad enviado por A, y A quiere estar seguro que el usuario con quien se está comunicando sea en realidad B. A y B también desean asegurarse que el contenido de sus mensajes enviados no sean alterados en el tránsito de los mismos. Por último A y B desean asegurarse que pueden establecer la comunicación (sin que haya una denegación de servicio por ejemplo). Dadas estas consideraciones, podemos identificar las siguientes propiedades deseables para una comunicación segura [2]:

- **Confidencialidad.** Sólo el usuario que envía y el usuario para el cual se encuentra destinado el mensaje pueden entender el contenido del mensaje. Para ello es necesario que el mensaje se encuentre revuelto u oculto (cifrado) de forma tal que no se entienda.

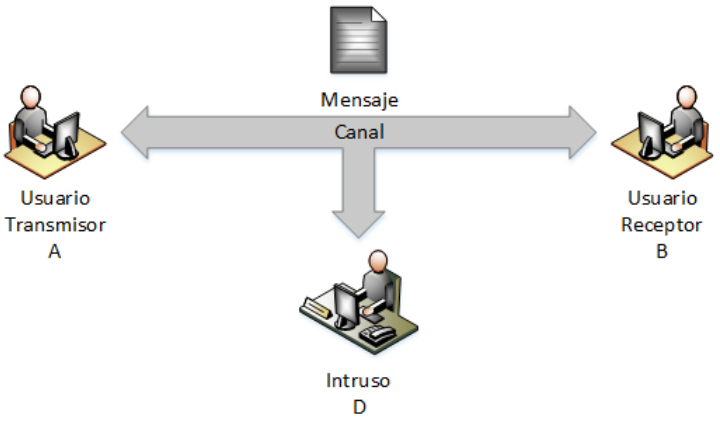

Figura 1.1: Comunicación entre dos usuarios con la intervención de un intruso

- **Integridad del mensaje.** Con esta propiedad ambos usuarios (el que envía y el que recibe) pueden confirmar que el mensaje no ha sido alterado ya sea por accidente o maliciosamente durante el tránsito del mismo.

- **Autenticación.** Ambos participantes de la comunicación pueden confirmar la identidad uno del otro.

- **Disponibilidad.** Asegurar que el sistema trabaje de forma apropiada y que el servicio no sea denegado a usuarios autorizados.

Entonces, para poder llevar acabo una comunicación de forma segura entre dos usuarios, se deberán presentar una o más de estas propiedades dependiendo del nivel de seguridad que se requiera. El establecimiento del nivel de seguridad necesario puede realizarse mediante un intercambio de mensajes de control que se efectúa con anterioridad al intercambio de datos. De no cumplirse ninguna de las propiedades mencionadas para una comunicación segura, un adversario podrá efectuar alguno de los ataques mencionados y obtendrá información que pueda afectar de manera significativa o catastrófica a los usuarios. Un ejemplo de ello sería que un adversario pueda conseguir la información de las cuentas bancarias de diversos usuarios.

Una vez que se tienen establecidas algunas de las definiciones más significativas y la necesidad de la seguridad en redes, es importante ahondar en otro tópico que constituye el punto de partida para poder llevar a cabo la confidencialidad, la integridad de los mensajes y la autenticación, que es la criptografía, la cual se explicará en la siguiente sección.

1.2. Principios de criptografía

La criptografía es considerada como el arte de escribir y resolver códigos. En sus inicios, los sistemas criptográficos tenían como un único objetivo poder establecer comunicaciones en secreto, disfrazando el mensaje (cifrado) de manera tal que solo los individuos que conocieran el código o algún secreto podían entenderlo. Actualmente, la criptografía cubre otros rubros como lo son la autenticación de mensajes, las firmas digitales, los protocolos para el intercambio de llaves, los protocolos de identificación de usuarios (autenticación), así como los problemas de ataques internos y externos que puedan surgir en los sistemas de cómputo distribuido. La criptografía moderna cuenta básicamente con dos paradigmas de cifrado (disfrazado o transformación de los mensajes), estos son los esquemas de cifrado simétrico, y los esquemas de cifrado asimétrico.

1.2.1. Criptografía de llave simétrica

La criptografía de llave simétrica (conocida también como criptografía convencional o de llave privada), fue la única conocida y utilizada antes del desarrollo de la criptografía de llave pública en los años setenta. Esta continúa siendo la más empleada debido a su relativamente bajo costo computacional y se centra en el uso de herramientas de sustitución y permutación.

La criptografía de llave simétrica contiene básicamente cinco elementos, los cuales se podemos apreciar en la figura 1.2:

- **Texto en claro:** Este es el mensaje original que se desea cifrar antes de ser enviado. Constituye una de las entradas del algoritmo de cifrado.
- **Llave Secreta:** Es otra de las entradas del algoritmo de cifrado. Es un conjunto de bits cuyo valor es independiente del texto en claro y del algoritmo de cifrado y que debe ser conocido sólo por las partes que se comunican.
- **Algoritmo de cifrado:** Lleva acabo una serie de sustituciones y transformaciones en el texto en claro con base en la llave utilizada. Diferentes llaves aplicadas a un mismo texto en claro producirán salidas diferentes, por lo que lo que dichas sustituciones y transformaciones del algoritmo dependen de la llave.
- **Texto cifrado:** Este es el mensaje transformado que se produce como salida del algoritmo. Su valor depende del texto en claro y de la llave secreta. El texto cifrado es aparentemente un flujo aleatorio de bits y por lo tanto no es entendible.
- **Algoritmo de descifrado:** Es prácticamente el inverso del algoritmo de cifrado. Toma como entradas el texto cifrado y la llave secreta y produce el texto en claro original.

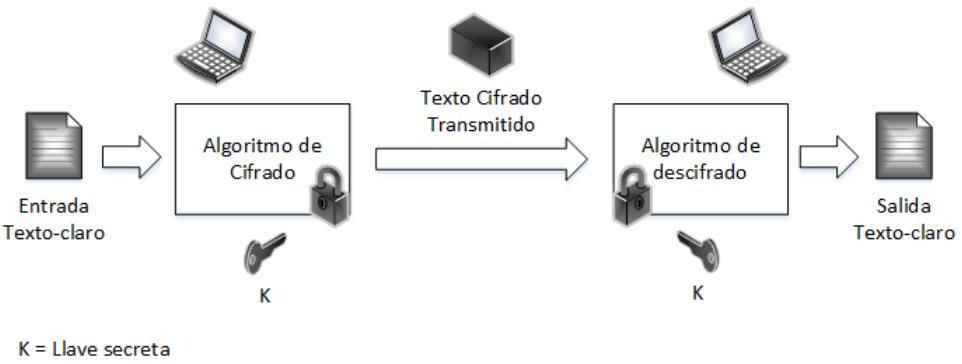

Figura 1.2: Modelo simplificado del esquema de cifrado simétrico

Existen dos requerimientos principales para el uso de las técnicas de cifrado simétrico:

1. Se requiere de un algoritmo de cifrado fuerte. Esto es: un adversario no deberá de ser capaz de descifrar el texto cifrado o de descubrir la llave a pesar de que tenga a su disposición pares de texto en claro/texto cifrado y el algoritmo de cifrado.

2. El usuario que envía y el usuario que recibe deben haber obtenido copias de la llave secreta de una forma segura y deben mantenerla protegida. Si un adversario puede obtener la llave por algún medio y tiene conocimiento del algoritmo, la comunicación entre ambas partes queda expuesta.

De acuerdo a los requerimientos mencionados, debido a que es inviable descifrar un mensaje a partir del texto cifrado y del conocimiento del algoritmo de cifrado, entonces no existe la necesidad de de mantener en secreto dicho algoritmo, por lo que toda la seguridad recae en poder mantener en secreto la llave de cifrado [3].

Atendiendo la forma en que la información se procesa, los algoritmos de cifrado se clasifican en dos grupos principales, los algoritmos de cifrado de bloques y los algoritmos de cifrado de flujo. Los algoritmos de cifrado de bloques cubren un mayor rango de aplicaciones, sobre todo en el área de cifrado simétrico. Entre los estándares más reconocidos y utilizados para los algoritmos de cifrado de bloques en la actualidad tenemos a DES (*Data Encryption Standard*) [4], su modificación más segura conocida como 3-DES (tres veces DES) y AES (*Advanced Encription Standard*) [5], el cual es el más seguro de los tres.

Típicamente, el objetivo de atacar un sistema de cifrado es poder recuperar la llave utilizada en lugar de recuperar sólo el texto en claro de un simple texto cifrado. Existen dos enfoques generales para atacar a un esquema de criptografía convencional:

- **Criptoanálisis:** Este tipo de ataque recae en la naturaleza del algoritmo y probablemente en algún conocimiento por parte del adversario de cierta información como lo son las características generales del texto en claro (propiedades probabilísticas por ejemplo) o pares de texto en claro/texto cifrado. Este tipo de ataque trata de explotar las características del algoritmo para intentar deducir un texto en claro específico o la llave utilizada para cifrarlo.

- **Ataque de fuerza bruta:** El atacante intenta cada posible llave sobre una pieza de texto cifrado hasta que obtenga un texto en claro inteligible o coherente. Por lo que para poder proporcionar seguridad en un sistema criptográfico, la longitud de llave debe ser tal que sea inviable, bajo cualquier poder de cómputo, obtenerla por éste método.

Uno de los problemas que se presenta en los sistemas de llave simétrica es la distribución segura de las llaves para comunicarse. Dado que para establecer una comunicación confidencial entre dos partes comunicantes se requiere que ambas tengan conocimiento de la misma llave sin que nadie más la conozca, entonces resulta lógico pensar que la llave debió haber sido entregada previamente a ambas partes de una forma segura.

Para solucionar este problema, existen diversas estrategias que pueden ir del caso más simple como lo es la distribución física de las llaves, es decir, que ambos usuarios se ven personalmente en algún lugar establecido por ellos, hasta la elaboración de diversos protocolos de distribución de las mismas, en los cuales, los individuos o entidades que se desean comunicar nunca tienen un contacto físico.

Otro inconveniente que se presenta en el modelo de llave simétrica es la administración de las llaves. Las llaves deben ser protegidas en los equipos pertenecientes a las partes comunicantes de tal forma que no puedan ser obtenidas por algún otro individuo malicioso. Además, si el medio de almacenamiento de un usuario es limitado, la cantidad de llaves que pueda almacenar para comunicarse con otros individuos también será limitada. Así si un usuario desea tener la posibilidad de comunicarse con N entidades diferentes, dado que requiere una llave por cada entidad con la que se desea establecer comunicación, entonces necesitará $N(N-1)/2$ llaves, lo cual nos pone en problemas con los requerimientos de memoria cuando la cantidad de usuarios es elevada.

Algo importante implicado en esto es que si la distribución de las llaves se lleva a cabo por medios presenciales, la distribución de las mismas resulta un caso totalmente no práctico si existen muchos usuarios con los que se desea poder comunicarse en un futuro. Para poder afrontar este problema (y el de espacio de almacenamiento si se presenta), existen estrategias de centralización en donde ciertas entidades denominadas centro de distribución de llaves

(KDC por sus siglas en inglés) administran y distribuyen llaves a los usuarios que desean establecer contacto de forma segura. Con ello, se logra que los usuarios sólo almacenen las llaves necesarias para comunicarse con dichas entidades centralizadoras y poder así obtener llaves que puedan utilizar para la comunicación con los usuarios deseados.

1.2.2. Criptografía de llave pública

La criptografía de llave pública surgió con la finalidad de resolver dos de los inconvenientes más importantes que se presentan en el sistema de llave simétrica: el problema de la distribución de llaves, y el problema de poder atribuirle un mensaje a un usuario (firmas digitales).

El esquema de cifrado de llave pública presenta seis elementos (figura 1.3):

- **Texto en claro:** Es el mensaje leíble que se utiliza como entrada en el algoritmo de cifrado.
- **Algoritmo de cifrado:** Lleva a cabo varias transformaciones en el texto en claro mediante el uso de una de las dos llaves relacionadas.
- **Llaves pública y privada:** Este es un par de llaves que han sido seleccionadas de tal manera que si una se utiliza para cifrar, la otra será utilizada para descifrar.
- **Texto cifrado:** Este es el mensaje transformado que se produce a la salida del algoritmo de cifrado. Este depende del texto en claro y de la llave utilizada para cifrar.
- **Algoritmo de descifrado:** Este algoritmo recibe como entradas el texto cifrado y la llave correspondiente para producir el texto en claro original.

Los pasos esenciales que sigue el esquema de llave asimétrico son los que se presentan a continuación:

1. Cada usuario genera un par de llaves relacionadas para ser utilizadas en el cifrado y descifrado de los mensajes.
2. Cada usuario coloca una de las dos llaves en un registro público que pueda ser accedido por los demás usuarios. A esta le denominaremos llave pública. A la otra llave se le conoce como llave privada y debe ser mantenida en secreto por el usuario que creo el par de llaves.
3. Si un usuario B desea enviar un mensaje confidencial a el usuario A, entonces B deberá cifrar el mensaje utilizando la llave pública de A.

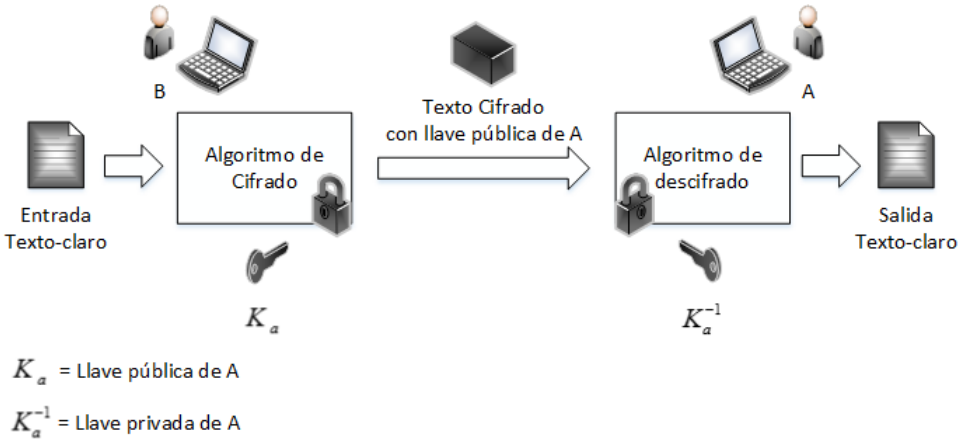

Figura 1.3: Modelo simplificado del esquema de cifrado asimétrico

4. Cuando A recibe el mensaje, lo descifra utilizando su llave privada. Nadie más puede descifrar el mensaje debido a que sólo A conoce su llave privada.

Con este enfoque, los participantes pueden tener acceso a las llaves públicas y poseer una colección de las mismas, mismas que utilizarán para comunicarse con cada una de las entidades existentes en el medio. Cada usuario deberá proteger y mantener en total secreto su llave privada con el fin de que se pueda tener una comunicación segura.

De acuerdo a la aplicación o a la manera en que se utilicen el par de llaves pública y privada para realizar alguna función criptográfica, podemos clasificar el uso de los sistemas de llave pública en tres categorías:

- **Cifrado/Descifrado:** El usuario que envía cifra un mensaje utilizando la llave pública del usuario que recibe el mensaje.
- **Firma digital:** El usuario que envía "firma" un mensaje con su llave privada. La firma se logra utilizando un algoritmo criptográfico aplicado al mensaje o a un pequeño bloque de datos que es función del mensaje.
- **Intercambio de llaves:** Dos lados cooperan para intercambiar una llave de sesión. Existen diferentes estrategias para hacerlo, envolviendo la llave o llaves privadas de una o las dos partes.

Entre los algoritmos de llave asimétrica más reconocidos y empleados en la actualidad tenemos el de Diffie-Hellman, Elgamal, RSA y Curvas Elípticas [1].

Al igual que los esquemas de llave simétrica, un esquema de llave pública es vulnerable a los ataques de fuerza bruta. Para contrarrestar este problema se lleva a cabo la misma medida que en los sistemas convencionales: utilizar llaves grandes. Dado que en los sistemas

de llave asimétrica se utilizan algunos tipos de funciones matemáticas (no invertibles), la complejidad para calcularlas varía en una escala no lineal de acuerdo al número de bits de la llave, por lo que entre más grande sea la llave, mayor será el procesamiento requerido para calcularlas. Entonces, debe existir un compromiso entre el nivel de seguridad requerido y el tiempo de cómputo necesario para calcular las funciones.

En la práctica, se han propuesto tamaños de llaves para hacer que el ataque de fuerza bruta sea inviable pero tienen el inconveniente de tener velocidades de cifrado/descifrado muy lentas para un uso de propósito general. Esto ha provocado que el uso de los sistemas de llave pública esté confinado a las aplicaciones de intercambio de llaves y firmas digitales.

1.2.3. Integridad de los mensajes

Como se mencionó, otra de las metas incluidas dentro de la criptografía es la integridad de los mensajes (o también conocida como autenticación de mensajes). Bajo esta meta se tienen dos objetivos principales: el primero, es garantizar que los mensajes no hayan sido alterados durante su transmisión por medios inseguros; y el segundo, es permitir al usuario que recibe un mensaje saber si dicho mensaje realmente fue originado por quien lo envió.

Para lograr estos objetivos, se hace uso de un tipo especial de funciones criptográficas, conocidas como "funciones *hash*". Una función *hash* toma como entrada un mensaje M, y calcula una cadena de bits de longitud fija $H(M)$ conocida como valor *hash*, que es, en un término sencillo, un comprimido del mensaje M. La principal característica que deben de tener estas funciones es que no debe ser posible (bajo cualquier poder de cómputo) obtener dos mensajes X y Y tal que su correspondiente valor *hash* sea el mismo ($H(X) = H(Y)$).

En otras palabras, un intruso no podrá sustituir un mensaje que sea protegido con la función *hash* por otro. Además de tener esta característica, las funciones *hash* tienen la cualidad de que requieren un bajo costo computacional que es inclusive menor al de los algoritmos de cifrado convencional, lo cual las hace muy útiles en la práctica. Entre los algoritmos *hash* más comunes (y empleados en la actualidad) tenemos el MD5 (RFC 1321 [6]) de Ron Rivest y el Algoritmo de Hash Seguro (SHA por sus siglas en inglés) en sus distintas versiones SHA-1, SHA-256, SHA-512, entre otras [7].

Regresando al problema de la integridad de los mensajes, un usuario A que desea enviar un mensaje a un usuario B podría calcular el valor *hash* del mensaje y enviárselo junto con el mensaje. Entonces B al recibir el mensaje llevaría a cabo el cálculo del valor *hash* del mensaje recibido y procedería a compararlo con el el valor *hash* recibido, dando por hecho que si ambos valores coinciden, el mensaje es íntegro. Pero en esto existe un problema. Si un adversario reemplaza el mensaje junto con su valor *hash* por otro mensaje distinto y le

añade el valor *hash* del nuevo mensaje, el usuario que recibe B no tendría forma de notar que el mensaje fue modificado ya que el valor *hash* recibido corresponde al de dicho mensaje alterado.

Entonces, para poder proporcionar la integridad de los mensajes, se recurre a utilizar un secreto compartido entre A y B en adición al uso de las funciones *hash* criptográficas. Dicho secreto no es más que una cadena de bits conocida como llave de autenticación. Utilizando este secreto, la autenticación de mensajes puede ser llevada a cabo como se describe a continuación.

1. A crea el mensaje M, lo concatena con el secreto S para formar $M + S$, y calcula un valor *hash* $H(M + S)$ al cual se le conoce como Código de Autenticación de Mensaje (MAC por sus siglas en inglés).

2. A agrega el MAC a el mensaje M, enviando $(M, H(M + S))$ a B.

3. B recibe el mensaje $(M, H(M + S))$ y conociendo S, calcula el MAC $H'(M + S)$. Si $H(M + S) = H'(M + S)$, entonces B concluye que todo está correcto.

En el ejemplo anterior se muestra una forma sencilla de generar MAC's, que es mediante la concatenación del mensaje con el secreto compartido y aplicando una función *hash* posteriormente. Existen en la actualidad estándares para generarlos, siendo el más popular el estándar HMAC (Hash MAC por sus siglas en inglés) que utiliza como bloques base alguna de las funciones *hash* comunes (MD5, SHA-1). HMAC se encuentra especificado en el RFC 2104 [8].

1.2.4. Firmas Digitales

Los esquemas de firmas digitales nos permiten plasmar la identidad de un usuario en un mensaje. Se basan en el uso de criptografía de llave asimétrica y operan como sigue:

1. Un usuario S establece un par de llaves pública (K_s) y privada (K_s^{-1}). La llave pública es puesta o distribuida de forma segura y es accesible para todos los usuarios.

2. El usuario S "firma" un mensaje M. Para ello utiliza un algoritmo de firma digital que tiene como entradas el mensaje y la llave privada K_s^{-1}, el cual realiza básicamente una operación de cifrado del mensaje con dicha llave. Si S envía el mensaje a un destinatario, entonces enviará dicho mensaje junto con su firma $(M, Sign_s(M))$.

3. Un usuario que reciba el mensaje $(M, Sign_s(M))$, empleará el algoritmo de verificación de firma digital que toma como entradas la firma $Sign_s(M)$ y la llave pública K_s

(llevando a cabo una operación de descifrado con esa llave) y proporciona a su salida un valor *M'*. Si $M = M'$, entonces el mensaje es correcto y proviene del usuario *S*.

Como se puede observar, un esquema de firmas digitales proporciona además de la identidad de quien crea el mensaje (autenticidad), integridad en el mismo. Estas características son muy similares a las de los códigos de autenticación de mensajes (MAC), pero a diferencia de estos, las firmas digitales presentan las siguientes ventajas:

- Simplifican el manejo de llaves. Si un usuario S desea enviar un mensaje M a varios participantes, sólo requerirá firmarlo una vez, con una única llave, que es en este caso, su llave privada. Si quisiera realizar lo mismo utilizando códigos MAC, el usuario S tendría que compartir un secreto con cada uno de los usuarios a los que les enviaría el mensaje.

- Son públicamente verificables. Esto significa que si un usuario recibe un mensaje firmado y lo verifica como legítimo (con el previo conocimiento de la llave pública de S), entonces es seguro que todos los usuarios que reciban este mensaje firmado lo verificarán como legítimo también.

- Poseen la propiedad de no repudio. Esta es una característica importante ya que con ella un usuario S que firma un mensaje no puede negar posteriormente haberlo hecho. Esto se debe a que S es el único individuo que posee su llave privada y nadie más podría generar la misma firma con cualesquiera otra llave.

La desventaja que presentan los esquemas de firma digital contra los códigos de autenticación de mensajes, es el tiempo de procesamiento, ya que requieren tiempos de dos o tres ordenes de magnitud mayores que los requeridos para la autenticación de mensajes mediante MAC debido al uso que hacen de los esquemas de cifrado asimétrico. Una alternativa que se utiliza en la práctica para disminuir los tiempos de procesamiento de las firmas digitales es calcular una función *hash* del mensaje que se desea enviar y firmar sólo el valor *hash* obtenido. Esto proporciona la ventaja de que el usuario sólo firmará una cadena de bits corta debido a la compresión realizada por la función *hash*.

Entre los algoritmos más comunes para firmas digitales tenemos a RSA, el Algoritmo de Firma Digital (DSA por sus siglas en inglés) y el Algoritmo de Firma Digital de Curvas Elípticas (ECDSA por sus siglas en inglés), los dos últimos incluidos en [9].

Una de las aplicaciones más importantes para las firmas digitales es la distribución de llaves públicas mediante el uso de "Certificados Digitales". En esencia un certificado consiste de una llave pública, un identificador del propietario de la llave y de una firma digital por parte de una tercera entidad confiable (Autoridad Certificadora) de toda la información

mencionada. La recomendación más utilizada en la práctica para la creación de certificados es el estándar ITU-T X.509. Por último podemos mencionar que al conjunto de hardware, software, personas, políticas, y procedimientos necesarios para crear, administrar, almacenar, y revocar certificados digitales basados en criptografía asimétrica se le conoce como "Infraestructura de Llave Pública" (PKI por sus siglas en inglés).

1.3. Autenticación de entidades

En esta sección se expondrán de manera detallada algunos de los principios elementales de autenticación de entidades en sistemas de cómputo distribuido. Además se describirán los modelos más comunes empleados para validar usuarios así como los posibles ataques que se pueden presentar durante el proceso de autenticación.

1.3.1. Principios

La autenticación de entidades (o usuarios) es el proceso en el cual dos entidades prueban su identidad una con la otra. Para lograrlo, una entidad deberá presentar cierta información (credenciales) para ser identificado y validado ante la entidad con la que desea establecer comunicación y de esta manera la entidad contactada podrá confiar en que la entidad que inició la conversación es quien dice ser. Adicionalmente, la entidad contactada podrá presentar sus credenciales a la entidad cliente para poder establecer así una relación de confianza mutua. De acuerdo a la definición del RFC 4949 [10], la autenticación consiste de dos pasos elementales:

- **Identificación:** Presentar un identificador (ID) al sistema de seguridad.
- **Verificación:** Presentar o generar información de autenticación y validar la misma de manera tal que corrobore el enlace entre la entidad y el identificador.

Mencionando un ejemplo para ilustrar lo anterior, un usuario o entidad A podría tener como identificador la palabra $ALFA$. Esta información necesita estar almacenada en algún servidor o sistema de cómputo el cual A deseara utilizar y puede ser conocida por el administrador del sistema y otros usuarios. Otra información relevante y típicamente utilizada para la autenticación es una contraseña asociada al identificador del usuario, que es conocida sólo por A y el sistema (no por otros usuarios). Para autenticarse, A deberá presentar la combinación identificador de usuario y contraseña al sistema. Una vez recibidos, el sistema procede al proceso de verificación y si resulta exitoso, se le otorgará acceso a la entidad A, de lo contrario, se le denegará el acceso.

Es importante mencionar que una característica que debe tener la contraseña mencionada en el ejemplo anterior, es que ningún otro usuario (además de A y el sistema) debería poder adivinarla u obtenerla por ningún medio. Si alguien más llegara a conocer la contraseña de A, dado que su identificador es conocido, podría suplantar la identidad de A y acceder a los recursos que se encuentran asignados para dicho usuario.

Como se puede notar (y como es en realidad en la mayoría de los contextos de seguridad), la autenticación es un bloque fundamental que constituye la primera línea de defensa y es la base para la mayoría de los tipos de control de acceso y manejo de cuentas de usuarios. La información que proporciona el usuario para autenticarse puede ser distinta (sin que quede limitada a sólo un ID y contraseña), y depende de cada sistema en particular. En términos generales, existen cuatro métodos de presentar información para autenticar la identidad de un usuario (los cuales pueden ser utilizados de manera independiente o en combinación) [1]:

- **Algo que el individuo conoce:** Ejemplos incluyen una contraseña, un número de identificación personal (PIN), o respuestas a un conjunto de preguntas preestablecidas.
- **Algo que el individuo posee:** Ejemplos incluyen llaves criptográficas, tarjetas de llave electrónica, tarjetas inteligentes y llaves físicas.
- **Algo que el individuo es:** Ejemplos incluyen reconocimiento por huella dactilar, retina y cara.
- **Algo que el individuo hace:** Ejemplos incluyen reconocimiento por patrón de voz, características de escritura de mano, y ritmo de escritura mecanográfica.

Todos éstos métodos, adecuadamente implementados y utilizados, pueden proveer una autenticación de usuarios segura. Sin embargo, cada método tiene sus problemas. Un adversario podría adivinar una contraseña que no sea lo suficientemente fuerte o robar alguna llave criptográfica de un equipo o sistema que no se encuentre bien protegido. Para la autenticación de entidades orientado a las redes de computadoras, los métodos más importantes no recurren a presentar información biométrica (algo que el individuo es y hace) sino más bien utilizan datos como lo son identificadores de usuario, contraseñas y llaves criptográficas que son enviados mediante intercambios de mensajes. A estos intercambios de mensajes se les conoce como "protocolos de autenticación" y generalmente son ejecutados antes de que las dos partes comunicantes corran algún otro protocolo. Dichos protocolos se basan en el uso de los sistemas criptográficos tanto de llave simétrica como de llave asimétrica para proteger (cifrar) ciertas partes de los mensajes.

En la literatura de los protocolos de autenticación es común encontrar a las entidades bajo el nombre de participantes (*principals* en inglés). De ahora en adelante nos referiremos

a los participantes de manera indistinta bajo ese término o bajo los términos de entidad o usuario. Cualquiera de los tres términos representarán ya sea a un dispositivo, un usuario humano o un proceso de cómputo y sólo se hará la distinción de que es lo que se interpreta con exactitud cuando sea necesario.

Una clasificación para la autenticación es en base a si se requiere que una o ambas entidades se identifiquen antes de comenzar sus operaciones, teniendo dos categorías:

- **Autenticación de una sola vía.** En esta, sólo una de las dos entiddes se autentica con la otra, por lo que la entidad que se identifica confía en la entidad con la que establece contacto. Un ejemplo se presentaba en las redes de telefonía celular de segunda generación GSM (*Global System for Mobile Communications*).En estas, el diseño de la red fue pensado de tal manera que el usuario podía confiar en los puntos de acceso (estaciones base) ya que era muy difícil para un adversario poder contar con la infraestructura necesaria para establecer su propio punto de conexión. De ahí que sólo se requería que los dispositivos móviles se autenticaran [11].

- **Autenticación mutua (o de dos vías).** Aquí, ambas partes que intervienen en el proceso de comunicación presentan material para autenticarse una con la otra. De esa manera, se puede establecer una relación de confianza mutua entre las dos entidades involucradas. Un ejemplo puede presentarse cuando un usuario en una computadora desea acceder a un servicio proporcionado por un servidor remoto; por un lado, el servidor deseará conocer la identidad del cliente para saber si podrá o no otorgarle el servicio, por el otro, el cliente también deseará saber si el servidor con quien se comunica es auténtico y no ha sido suplantado por otro servidor de algún adversario que desee robar información.

1.3.2. Modelos de autenticación

De acuerdo a el número de entidades participantes en el proceso de autenticación, se tienen dos modelos distintos [12]:

- **De dos participantes.** En este modelo dos entidades interactúan una con la otra sin la intervención de alguna entidad intermedia que los ayude a llevar a cabo el proceso de autenticación. Es básicamente una comunicación directa entre un cliente y un servidor.

- **De tres participantes.** Para este modelo existe una tercera entidad la cual interviene durante el procedimiento de autenticación. Existen muchas estrategias para este tipo de modelo. Se describirán algunas de ellas durante el desarrollo de las siguientes secciones y capítulos.

1.3.3. Ataques a los protocolos de autenticación

Los protocolos de autenticación se encuentran expuestos a diversos problemas, entre los cuales destacan principalmente los problemas de confidencialidad y los de temporización. Atendiendo al problema de confidencialidad, debido a que los protocolos de autenticación se basan en el uso de sistemas criptográficos para proteger algunas partes de los mensajes, requieren que estos sistemas (ya sean simétricos o asimétricos) sean lo suficientemente fuertes para prevenir el robo de identidades mediante el uso de llaves comprometidas obtenidas a partir de técnicas de criptoanálisis o ataques de fuerza bruta. Con respecto al problema de temporización, podemos decir que resulta muy importante poder establecer relaciones temporales en los mensajes intercambiados con la finalidad de poder evitar los ataques de repetición de mensajes (replay attacks en inglés).

Es común que a la hora de diseñar o analizar un protocolo de autenticación se elija un sistema criptográfico que sea lo suficientemente seguro para el propósito del mismo, dejando como tarea de diseño, buscar estrategias de cómo utilizar dichos esquemas criptográficos para lograr el reconocimiento de los usuarios sin que la repetición de mensajes pueda tener un efecto adverso. Tales repeticiones de mensajes, en el peor de los casos, permitirán a un adversario comprometer alguna llave criptográfica o suplantar de manera exitosa la identidad de algún otro usuario. Como mínimo, una repetición exitosa podrá interrumpir o alterar las operaciones realizadas por los participantes mediante la presentación de material (por parte del adversario) que parezca genuino pero que en realidad no lo sea. Gong [13] menciona los siguientes ejemplos de ataques de repetición:

1. El ataque más simple de repetición es aquel en el que el oponente simplemente copia un mensaje y lo replica después.

2. Un oponente puede replicar un mensaje que posee una estampa de tiempo cuyo valor se encuentra dentro de una ventana de tiempo válida. Si el mensaje original y el mensaje replicado llegan dentro de la ventana de tiempo, este hecho puede ser detectado (y posiblemente registrado en alguna bitácora).

3. Como en el ejemplo dos, un oponente puede replicar un mensaje que contiene una estampa de tiempo que se encuentra dentro de una ventana de tiempo válida, pero adicionalmente, el oponente suprime el mensaje original. En este caso, la repetición no puede ser detectada.

4. Otro ataque involucra una repetición hacia atrás del mensaje sin hacerle alguna modificación. Esto es, regresar de vuelta el mensaje al usuario que lo envió. Este ataque es

posible si se utiliza cifrado de llave simétrica y la entidad que envía el mensaje no tiene manera de diferenciar entre los mensajes enviados y recibidos en base a su contenido.

Básicamente, existen dos enfoques generales con los que se le hace frente a los ataques de repetición, los cuales son:

- **Estampas de tiempo:** Un usuario A acepta un mensaje como reciente sí y sólo sí el mensaje contiene una estampa de tiempo que, en el juicio de A, es lo suficientemente cercana al valor del tiempo actual conocido por A. Este enfoque requiere que los relojes entre los participantes se encuentren sincronizados.

- **Reto/Respuesta:** Un usuario A, en espera de un mensaje reciente por parte de B, envía primero a B un reto (que por lo común es una cadena de bits de valor aleatorio conocida en la literatura bajo el nombre de *nonce*) y espera una respuesta por parte de B que contenga el valor correcto enviado previamente por A. El reto enviado por A puede ser modificado por B utilizando una función conocida entre ambos.

Cada una de estas estrategias presentan sus ventajas y desventajas, por lo que se debe ser cuidadoso para emplearlas de manera prudente. Para las estampas de tiempo como se mencionó, se requiere de sincronización, lo cual es una tarea difícil considerando la naturaleza impredecible de los retardos en las redes, de ahí que se utilicen ventanas de tiempo. En el caso de los retos, es necesario que el adversario no pueda predecir dichos valores, de lo contrario podrá efectuar diversos ataques de repetición.

1.4. Autenticación de entidades mediante el uso de criptografía simétrica

En esta sección daremos un vistazo a algunos de las primeras propuestas de protocolos de autenticación basados en la criptografía de llave simétrica que sentaron las bases para muchos de los trabajos posteriores. Antes de introducir dichos protocolos, es necesario mencionar algunos conceptos adicionales relacionados a las llaves criptográficas así como la notación que se empleará para describirlos.

Como se mencionó, las llaves criptográficas son utilizadas por los algoritmos de cifrado para transformar la información de tal manera que no pueda ser entendida ni revertida a menos que se tenga conocimiento de las mismas. Dependiendo de si estas llaves son de corta o larga duración podemos clasificarlas de dos formas:

- **Llaves de sesión:** Este tipos de llaves tienen un tiempo de vida relativamente corto como puede ser algún par de horas o una conexión a un servidor. Una vez que transcurra el tiempo de duración establecido o se termine la conexión, si dos entidades desean comunicarse de nuevo deberán utilizar una nueva llave independiente de la anterior. Esta estrategia le hace frente a los ataques de llaves comprometidas o de criptoanálisis. Son empleadas para proteger la información de los usuarios durante los intercambios de mensajes posteriores al protocolo de autenticación.

- **Llaves maestras:** Son llaves de larga duración que son utilizadas por lo general para proteger la distribución de las llaves de sesión. Se busca utilizarlas lo menos posible en los intercambios de mensajes de algún protocolo de autenticación para que no sean sometidas a criptoanálisis y deben estar lo suficientemente protegidas por los usuarios propietarios de las mismas. Las llaves maestras deben ser cambiadas cuando ya ha pasado un tiempo considerablemente grande (que es mucho mayor al de las llaves de sesión) o cuando se encuentren comprometidas.

De esta clasificación de las llaves criptográficas, podemos establecer una jerarquía de tal forma que en el nivel más alto se encuentran las llaves maestras, y de ahí las llaves de sesión. La distribución de las llaves maestras es por lo general por medios físicos, pero se puede llevar a cabo una estrategia en la cual se utiliza la criptografía asimétrica para realizar esta tarea. Esta táctica, si se implementa, añade un nivel más a la jerarquía siendo las llaves pública y privada el nivel más alto.

Con respecto a la notación que se utilizará para describir protocolos de autenticación, las entidades involucradas serán simbolizas utilizando las letras mayúsculas del alfabeto español como puede ser A, B, S o mediante una agrupación de dos o más letras mayúsculas como por ejemplo AS, KDC. Relacionado a los elementos que componen a los mensajes, tenemos la siguiente notación:

- ID_x representa el identificador de un usuario X. Entonces, ID_a representa el ID del usuario A.

- $K_{x,y}$ simboliza a una llave compartida sólo por los usuarios X y Y. $K_{a,b}$ será una llave que sólo conocerán los usuarios A y B y que utilizarán para comunicarse de forma confidencial.

- N_x representa un *nonce* emitido por el usuario X. Por lo tanto, N_a es un *nonce* dicho por el usuario A.

- T_x simboliza a una estampa de tiempo generada por el usuario X. T_a será una estampa de tiempo generada por el usuario A.

- $\{M\}_{Kx,y}$ representa a un mensaje M cifrado con la llave $K_{x,y}$.

Los mensajes por lo general se encuentran constituidos por una concatenación de varios mensajes pequeños, tal concatenación la expresaremos como sigue:

$$M = ID_a, ID_b, N_a$$

Para representar el sentido de los mensajes, se utilizará el símbolo "$\rightarrow$". La representación $A \rightarrow B$ indica que el usuario A le envía un mensaje al usuario B. Con esta notación será suficiente para la mayoría de los protocolos de autenticación que empleen el esquema de cifrado simétrico. La notación adicional que se requiera para explicar algunos protocolos será introducida junto con ellos.

1.4.1. Protocolos de autenticación de llave simétrica

Una de las primeras propuestas es el protocolo de autenticación de Needham y Schroeder [14]. Este protocolo se basa en el uso del modelo de tres entidades, en el cual un tercer participante denominado centro de distribución de llaves (KDC por sus siglas en inglés) es el responsable de generar y distribuir llaves de sesión para ser utilizadas a través de una conexión entre dos entidades. En la figura 1.4 se presenta el intercambio de mensajes en el protocolo.

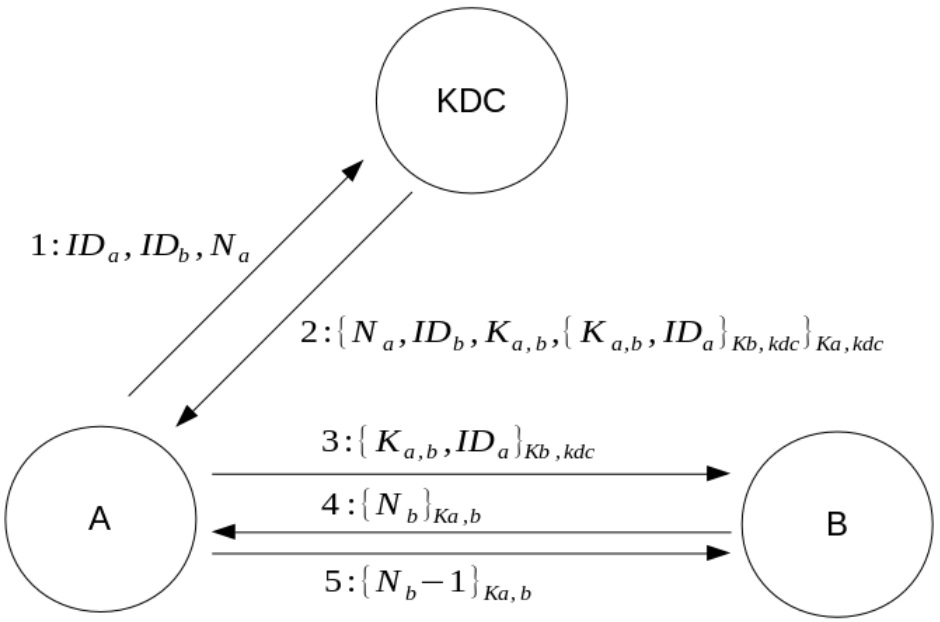

Figura 1.4: Protocolo de autenticación de Needham-Schroeder

Las acciones realizadas en cada paso se describen a continuación:

1. El protocolo inicia con el usuario A enviándole un mensaje al servidor KDC. En este mensaje, A le informa al KDC que desea establecer una comunicación con el usuario B mediante el uso de los identificadores ID_a e ID_b. Además le envía un *nonce* (N_a) en espera de que la respuesta del KDC lo incluya para verificar que esta sea reciente. El mensaje se envía sin cifrar.

2. Una vez recibida la petición, el KDC en primer lugar valida si esos usuarios existen en el sistema y luego genera una llave de sesión $K_{a,b}$ que será utilizada por A y B para comunicarse. El KDC responde un mensaje que incluye la llave $K_{a,b}$, el *nonce* N_a e información adicional que A utilizará para validar el mensaje y para posteriormente poder dirigirse con B. El KDC envía el mensaje cifrado con la llave $K_{a,kdc}$ para protegerlo contra adversarios.

3. A recibe el mensaje, lo descifra y verifica que la información recibida sea la esperada. Además de la llave de sesión $K_{a,b}$, A recibe un conjunto de datos cifrados con la llave $K_{b,kdc}$ (al que denominaremos como *ticket*), que sólo B podrá leer. A envía el *ticket* a B.

4. B recibe el *ticket* y extrae los datos contenidos. Este *ticket* le dice a B que el usuario A desea establecer contacto con el y para comprobar que este usuario es quien dice ser le envía un reto (N_b) cifrado con la llave de sesión que obtuvo del *ticket*.

5. A recibe el reto cifrado con $K_{a,b}$, le aplica una función (en este caso de resta) y se lo regresa a B. Con ello tanto A como B (cuando reciba el reto modificado) podrán establecer una relación de confianza mutua y además se encontrarán en posesión de una llave criptográfica que sólo ellos conocen para proteger futuros intercambios de datos.

A pesar del acuerdo establecido en los pasos 4 y 5, este protocolo presenta un problema. Suponiendo que un oponente ha podido comprometer alguna llave de sesión utilizada con anterioridad, éste podrá reenviar el mensaje del paso 3 a B y suplantar la identidad de A. Si B no tiene un registro de las llaves previamente utilizadas, no tendrá manera de darse cuenta si el *ticket* recibido es o no una repetición.

Una de las alternativas que surgió para resolver el inconveniente presentado el protocolo de Needham-Schroeder fue realizada por Denning y Sacco en [15]. En su propuesta, se incluye el uso de estampas de tiempo en los pasos dos y tres del protocolo de Needham-Schroeder para garantizar la frescura de la llave de sesión. De inicio se hace la suposición de que los relojes de las tres entidades que participan en la comunicación se encuentran sincronizados. El intercambio de mensajes es el siguiente:

1. $A \rightarrow KDC: \quad ID_a, ID_b$
2. $KDC \rightarrow A: \quad \{K_{a,b}, ID_b, T_{kdc}, \{K_{a,b}, ID_a, T_{kdc}\}_{Kb,kdc}\}_{Ka,kdc}$
3. $A \rightarrow B: \quad \{K_{a,b}, ID_a, T_{kdc}\}_{Kb,kdc}$
4. $B \rightarrow A: \quad \{N_b\}_{Ka,b}$
5. $A \rightarrow B: \quad \{N_b + 1\}_{Ka,b}$

En el intercambio, T_{kdc} es la estampa de tiempo generada por el *KDC*. Tanto el usuario A como el usuario B pueden verificar el valor de T_{kdc} considerando un rango o ventana de tiempo. Esto debido a las ligeras discrepancias que puedan existir entre sus relojes así como a los retardos generados en la red.

Gracias a esta estrategia, un adversario que conozca alguna llave de sesión comprometida y decida replicar el mensaje tres del protocolo tendrá menores oportunidades de poder engañar al usuario B. La posibilidad de que un adversario pueda tener éxito en hacerle creer a B que la llave es reciente depende de la ventana de tiempo empleada por este usuario. Si la ventana de tiempo es grande, el adversario tendrá más espacio para poder replicar el mensaje.

A pesar de que el protocolo de Denning-Sacco parece proveer un mayor grado de seguridad, no queda libre de ataques. Dado que se utilizan relojes sincronizados en una red distribuida, existe el riesgo de que los relojes pierdan su sincronía como resultado de un sabotaje al sistema o que el mecanismo de sincronía falle. El problema se genera cuando el reloj del usuario que envía (en este caso A) se encuentra adelantado con respecto al del usuario que recibe (B). En este caso, un adversario puede interceptar el mensaje enviado por A y replicarlo después cuando el la estampa de tiempo llegue a tener el valor del tiempo actual de B. Este ataque es conocido como ataque de supresión-repetición.

Existen alternativas para contrarrestar los ataques de supresión-repetición. Una de ellas es hacer que los participantes se vean forzados a verificar frecuentemente sus relojes contra el reloj del *KDC*. Otra es buscar la manera de no depender de relojes sincronizados, tal vez por medio del uso de nonces. A continuación se presenta un protocolo propuesto en [16] que busca evitar este tipo de ataque utilizando nonces:

1. $A \rightarrow B: \quad ID_a, N_a$
2. $B \rightarrow KDC: \quad ID_b, N_b, \{ID_a, N_a, T_b\}_{Kb,kdc}$
3. $KDC \rightarrow A: \quad \{ID_b, N_a, K_{a,b}, T_b\}_{Ka,kdc}, \{ID_a, K_{a,b}, T_b\}_{Kb,kdc}, N_b$
4. $A \rightarrow B: \quad \{ID_a, K_{a,b}, T_b\}_{Kb,kdc}, \{N_b\}_{Ka,b}$

Siguiendo el protocolo paso a paso:

1. A inicia la comunicación con B enviándole su identificador (ID_a) y un *nonce* (N_a) en espera de que este sea respondido posteriormente.

2. B le informa al KDC que requiere de una llave de sesión. En su mensaje B le envía su identificador junto con un *nonce* (N_b) que utilizará más adelante para verificar que un mensaje sea reciente. El mensaje de B además incluye un bloque de datos cifrado con la llave que comparte con el KDC. Estos datos son utilizados con el objetivo de indicarle al KDC que emita credenciales para el usuario A además de sugerir un tiempo de expiración de las mismas.

3. El KDC le proporciona a A el *nonce* generado por B así como dos bloques de información cifrada. El primer bloque se encuentra cifrado con la llave que comparten A y KDC, en éste el KDC le informa a A que B recibió recientemente el *nonce* N_a por lo que la llave de sesión $K_{a,b}$ contenida en el mismo es válida y con cierta vigencia (gracias a T_b). El segundo bloque se encuentra cifrado con la llave que comparten B y KDC, y será utilizado por A como *ticket* para subsecuentes autenticaciones.

4. A transmite el ticket a B junto con el *nonce* de B cifrado con la llave de sesión $K_{a,b}$. El *ticket* provee a B la llave secreta que utilizará para recuperar el *nonce* N_b. Con esto B tiene conocimiento de que el mensaje proveniente de A no es una repetición.

Este protocolo proporciona una manera efectiva y segura de establecer llaves de sesión entre dos entidades.

En el capítulo tres estudiaremos un protocolo de autenticación muy efectivo basado en el protocolo de Needham-Schroeder que es un estándar de la IETF (*Internet Engineering Task Force*) ampliamente utilizado en la actualidad. Es conocido bajo el nombre de Kerberos y fue desarrollado en el MIT (*Massachusetts Institute of Technology*).

1.5. Autenticación de entidades mediante el uso de criptografía asimétrica

En esta sección daremos una revisión a algunas propuestas de protocolos de autenticación que emplean los sistemas de criptografía de llave asimétrica. Como se mencionó, los sistemas de criptografía de llave pública son muy costosos en cuestión de procesamiento computacional. Es por ello que la mayoría de los protocolos de autenticación que los emplean buscan, además de autenticar a los usuarios involucrados realizando la menor cantidad de operaciones posibles, generar y distribuir de forma segura una llave de sesión simétrica que los usuarios puedan utilizar para futuros intercambios de información. Para comenzar, se introducirá un poco más de notación que complementa a la notación mencionada previamente.

En relación a la notación adicional, tenemos lo siguiente:

- K_a y K_a^{-1} representan el par de llaves pública y privada respectivamente de un usuario A.

- $Cert_A$ corresponde al certificado digital de la entidad A así como $Cert_B$ corresponde al certificado digital de la entidad B.

- $[M]_{Ka^{-1}}$ representa un mensaje M firmado digitalmente con la llave privada del usuario A.

Hemos visto que uno de los objetivos de algunos de los protocolos de autenticación es, además de identificar y validar un usuario, establecer una llave de sesión que las entidades puedan utilizar para futuras comunicaciones. De acuerdo a la forma en que la llave de sesión es establecida, los protocolos de autenticación pueden ser llamados de dos maneras:

- *Protocolos de distribución de llaves autenticado.* En estos, la llave es generada y distribuida por uno de los participantes del proceso de autenticación o por una tercera entidad en la que los participantes confíen (como puede ser un centro de distribución de llaves).

- *Protocolos de acuerdo de llave autenticado.* Para este tipo de protocolos, las entidades que participan en el proceso de autenticación colaboran en la generación de una llave de sesión. Esta llave puede ser construida a partir de la aplicación de algoritmos matemáticos que utilicen números aleatorios generados por las entidades en cuestión.

El intercambio de llaves de Diffie-Hellman [17] es un algoritmo que es empleado como punto de partida en los protocolos de acuerdo de llave autenticado. El propósito de este algoritmo es permitir que dos usuarios intercambien de manera segura una llave que puedan utilizar para el cifrado simétrico de mensajes subsecuentes. Una explicación detallada del intercambio de llaves de Diffie-Hellman se puede encontrar en [1].

1.5.1. Protocolos de autenticación de llave asimétrica

El siguiente protocolo fue presentado por Woo y Lam en [18]. Consiste de un intercambio de mensajes en el que interviene un centro de distribución de llaves (KDC) y se presenta a

continuación:

1. $A \rightarrow KDC:$ ID_a, ID_b
2. $KDC \rightarrow A:$ $[ID_b, K_b]_{Kkdc^{-1}}$
3. $A \rightarrow B:$ $\{N_a, ID_a\}_{Kb}$
4. $B \rightarrow KDC:$ $ID_a, ID_b, \{N_a\}_{Kkdc}$
5. $KDC \rightarrow B:$ $[ID_a, K_a]_{Kkdc^{-1}}, \{[N_a, K_{a,b}, ID_a, ID_b]_{Kkdc^{-1}}\}_{Kb}$
6. $B \rightarrow A:$ $\{N_b, [N_a, K_{a,b}, ID_a, ID_b]_{Kkdc^{-1}}\}_{Ka}$
7. $A \rightarrow B:$ $\{N_b\}_{Ka,b}$

Antes de comenzar el intercambio tanto el usuario A como el usuario B deben conocer la llave pública del centro de distribución de llaves, que es a su vez una entidad certificadora. En el primer mensaje, A le informa a el KDC su intención de comunicarse con B. El KDC le proporciona a A la llave pública de B en el mensaje dos. A se dirige hacia B mandándole un reto (N_a) junto con su identificador cifrados con la llave pública de B (mensaje tres). Al recibir la petición de A para establecer una comunicación, B envía el mensaje cuatro en el cual cifra N_a con la llave pública del KDC. El KDC le proporciona a B (en el mensaje cinco) la llave pública de A junto con un conjunto de datos firmados por él ($N_a, K_{a,b}, ID_A, ID_B$). Esta información básicamente le dice a B que $K_{a,b}$ es una llave de sesión secreta generada por el KDC y que ocupará para futuros intercambios de mensajes con A. B le envía los datos firmados por el KDC a A junto con un reto N_b en el mensaje seis. Cuando A recibe el mensaje, extrae los datos utilizando su llave privada y la llave pública del KDC y confirma que la llave de sesión recibida es reciente gracias al reto N_a. A responde al reto hecho por B enviando N_b cifrado con la llave de sesión adquirida (mensaje siete). Con esto, B tiene la certeza de que A se encuentra activo en ese momento y que recibió de manera satisfactoria la llave $K_{a,b}$.

Este parece ser un protocolo seguro que toma en cuenta varios ataques. A pesar de ello presenta la desventaja de que se requieren muchos intercambios de mensajes en los cuales se llevan a cabo operaciones de cifrado asimétrico.

A continuación se presenta un protocolo basado en el intercambio de llaves de Diffie-Hellman conocido como protocolo STS (*Station to Station* por sus siglas en inglés). Para la operación del protocolo, se tiene un par de números "p" (número primo apropiado) y "g" (elemento primitivo perteneciente al campo finito *GF(p)*) que son fijos y públicamente conocidos. Una explicación a detalle acerca de números primos y campos finitos se puede encontrar en [1]. Las entidades participantes A y B utilizan un esquema común de firmas establecido por la autoridad certificadora T, por lo que confían y conocen la llave pública de esta entidad. El intercambio de mensajes es el siguiente:

1. $A \rightarrow B: \quad Y_a$
2. $B \rightarrow A: \quad Y_b, Cert_B, \{[Y_b, Y_a]_{Kb^{-1}}\}_{Ka,b}$
3. $A \rightarrow B: \quad Y_a, Cert_A, \{[Y_a, Y_b]_{Ka^{-1}}\}_{Ka,b}$

Aquí, $Cert_A = (ID_a, K_a, [ID_a, K_a]_{Kt^{-1}})$ y $Cert_B = (ID_b, K_b, [ID_b, K_b]_{Kt^{-1}})$. Las acciones realizadas son:

1. A genera un número entero positivo aleatorio X_a, calcula $Y_a = g^{X_a}$ y envía Y_a a B.

2. Una vez que lo recibe, B genera un número entero positivo aleatorio X_b, calcula $Y_b = g^{X_b}$ y la llave de sesión $K_{a,b} = (Y_a)^{X_b}$. Además, B calcula la firma de autenticación $[Y_b, Y_a]_{Kb^{-1}}$ y le envía a A la firma cifrada con la llave de sesión $K_{a,b}$ junto con su certificado.

3. A recibe la información y con Y_b calcula la llave de sesión $K_{a,b} = (Y_b)^{X_a}$. A verifica la validez del certificado obtenido y si es válido obtiene la llave pública K_b. Empleando la llave $K_{a,b}$, A obtiene la firma de autenticación de B y procede a verificarla con la llave K_b. Si los valores obtenidos en la firma son los esperados, entonces A le responde a B un mensaje que contiene su certificado y su firma de autenticación $[Y_a, Y_b]_{Ka^{-1}}$ protegida con la llave de sesión generada. Al recibir el mensaje, B procede de manera análoga y si los datos recibidos son correctos, ambas entidades establecen una relación de confianza.

Como se puede ver en este protocolo, se resuelve el problema presentado en el intercambio de llaves de Diffie-Hellman y una vez ejecutado ambas entidades tendrán la confianza de que cuentan con una llave de sesión segura (que nadie más conoce) para comunicaciones posteriores. El protocolo STS es un buen protocolo debido a que además de tener un intercambio de mensajes breve, cumple con varios de los requerimientos necesarios para la autenticación de usuarios (los cuales serán descritos con mayor detalle en el capítulo tres). De manera similar al protocolo Woo y Lam, presenta la desventaja de requerir de varias operaciones costosas por lo que no es muy conveniente utilizarlo en dispositivos con relativamente poco poder de procesamiento.

1.6. Propiedades de seguridad de los protocolos de autenticación

Como se ha visto con los ejemplos de autenticación presentados en la sección anterior, los objetivos de todo protocolo de autenticación consta de dos cosas: la primera es el proceso

de autenticación como tal (identificación y verificación de las entidades participantes) y la segunda se trata del establecimiento de una llave de sesión. En documentos de la IETF, las propiedades de seguridad usuales de los protocolos de autenticación se basan en la integración de las metas de estos dos objetivos, los cuales se pueden dividir a su vez en seis sub-metas [19] tal como se muestra en la figura 1.5.

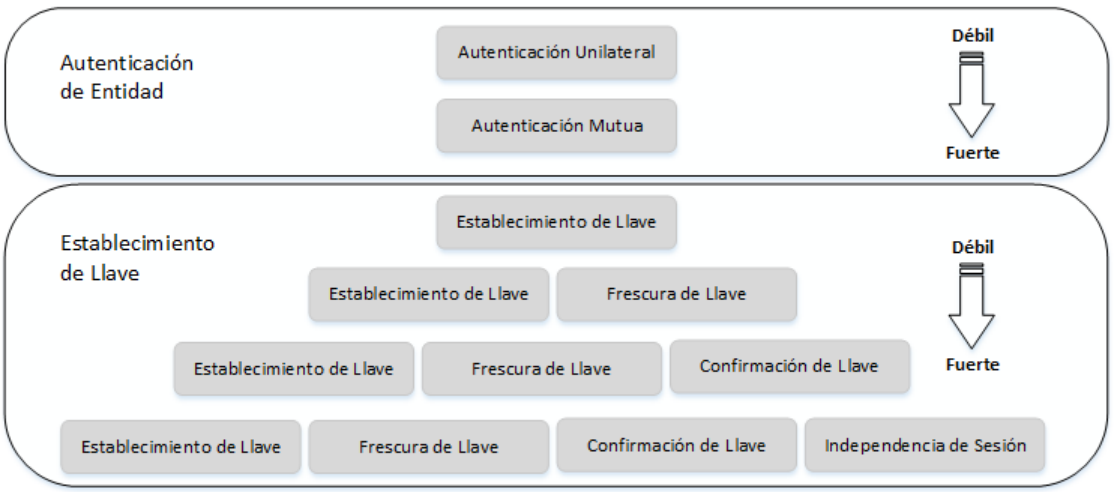

Figura 1.5: Propiedades de seguridad para los protocolos de autenticación

La autenticación de entidades, como se ha mencionado a lo largo de este capítulo, tiene el objetivo de verificar que las entidades participantes sean quienes dicen ser. La figura 1.5 indica claramente el hecho de que establecer la autenticación mutua es una medida de seguridad más fuerte que sólo establecerla para una entidad.

Las propiedades del acuerdo de llaves consisten del establecimiento de la llave (de sesión), la frescura de la llave, la confirmación de ésta y la independencia de sesión. El establecimiento de la llave de sesión significa que una llave de sesión es generada y compartida por ambas entidades al final del intercambio de mensajes. La frescura de llave simplemente requiere de que las entidades participantes se aseguren que la llave de sesión generada sea nueva. La confirmación de llave tiene el objetivo de describir escenarios en donde ambos participantes comprueban la existencia del uno con el otro mediante el uso (en alguno de los mensajes del protocolo) de la llave de sesión recién generada. Finalmente, un protocolo es considerado con "independencia de sesión" si cualesquiera llaves de sesión comprometidas no proporcionan alguna pista acerca de las futuras llaves de sesión.

De igual forma que en el rubro de la autenticación de entidades, entre más propiedades del acuerdo de llaves se cumplan, mayor seguridad proporciona el protocolo de autenticación. Combinando todas estas características dentro de un protocolo de autenticación, se puede obtener una mayor seguridad dentro del mismo.

Capítulo 2

Herramientas para la evaluación y diseño de los protocolos de autenticación

En este capítulo, se estudian algunas de las herramientas de evaluación y diseño de los protocolos de autenticación en ambientes distribuidos. Se hablará en un comienzo acerca de las herramientas de análisis lógico propuestas por algunos autores con la finalidad de formalizar el estudio de los protocolos de autenticación. Posteriormente se explican algunos principios relacionados al diseño correcto de los protocolos autenticación. Finalmente se mencionan algunas consideraciones de caracter práctico que deben ser tomadas en cuenta a la hora de realizar un diseño de un protocolo de autenticación.

2.1. Herramientas de Análisis lógico en los protocolos criptográficos

La verificación formal de los protocolos de autenticación se ha utilizado ampliamente para comprobar de manera formal si un protocolo cumple con algunos de los objetivos necesarios para garantizar autenticación. En 1989, se propuso el primer método formal para el análisis de protocolos de autenticación desarrollado por Burrows, Abadi y Needham, el cual fue nombrado como lógica BAN [20]. BAN es una lógica basada en la lógica modal de las creencias y utiliza fórmulas (sentencias a las cuales se les puede atribuir un valor de verdad o falsedad) para expresar las relaciones de creencia entre las entidades participantes. La lógica BAN es simple (en cuestión de notación y lógica) y útil para revelar algunos de los fallos en los protocolos de autenticación. Sin embargo, cuenta con una serie de limitaciones semánticas que propiciaron algunas propuestas para mejorarla. Por ejemplo, Abadi y Tuttle (lógica AT) establecieron una semántica más adecuada en su extensión de BAN [21]. Gong, Needham,

y Yahalom en [22] (lógica GNY) agregaron y replantearon las reglas de BAN para obtener un mejor razonamiento de las relaciones de creencia entre las entidades participantes. Van Oorschot (Lógica VO) agregó más reglas para razonar acerca de los protocolos de acuerdo de llaves [23]. En 1994, Syverson y Van Oorschot desarrollaron la lógica SVO [24] con el objetivo de unificar la familia de lógicas derivadas a partir de BAN (GNY, AT, VO). SVO es una lógica que a pesar de englobar a las demás lógicas, no resulta ser muy extensa ni mucho menos difícil de emplear a comparación de sus predecesoras, por lo que es una herramienta atractiva para el análisis de los protocolos de autenticación. En esta sección se describirán los fundamentos necesarios para el uso y entendimiento de la lógica SVO. No se explicarán detalles relacionados a lenguajes formales y lógica, por lo que se puede consultar [25] y [26] para una mejor comprensión.

2.1.1. Lógica SVO

La lógica SVO fue elaborada por Paul F. Syverson y Paul C. Van Oorschot y presentada por primera en vez en [24]. Se trata de una lógica modal que surgió con la idea de cubrir algunas de las limitaciones de la lógica BAN (y de sus extensiones) presentando un modelo de computación y una lógica solvente con respecto a ese modelo sin perder la expresividad de sus predecesoras. Se basa en el uso de axiomas y reglas de inferencia para interpretar las metas en los protocolos criptográficos.

En SVO, los protocolos son analizados de manera abstracta, por lo que los mensajes que se envían se representan mediante expresiones compuestas de un lenguaje en lugar de sólo cadenas de bits. Además, se considera que el utilizar cifrado garantiza que cada parte cifrada no podrá ser alterada o ser construida a partir de pequeñas partes de información, por lo que los sistemas de cifrado se consideran seguros dentro del análisis formal.

Al igual que en el análisis BAN, SVO hace distinción entre dos épocas: el pasado y el presente. La época presente comienza al inicio de una ejecución particular del protocolo bajo consideración. Todos los mensajes enviados con anterioridad a este tiempo se consideran que se encuentran en el pasado. Un protocolo de autenticación debe de ser lo suficientemente cauteloso para percatarse de aquellos mensajes del pasado y de no aceptarlos en la ejecución actual del mismo.

2.1.2. Notación SVO

En la lógica SVO, las entidades participantes son nombradas mediante letras en mayúscula como A, B, P, Q, S. Para la representación de fórmulas lógicas arbitrarias, se emplean los símbolos metalingüísticos φ y ψ; y para la representación de mensajes, se utilizan las

letras X y Y. Se tiene además la siguiente notación adicional:

- P *believes* X: El participante P actúa considerando que el mensaje X es cierto [27].
- P *received* X: El participante P ha recibido un mensaje que contiene X. P puede leer y replicar X.
- $\neg\varphi$: Niega la fórmula φ.
- P *said* X: El participante P envió un mensaje que contiene X.
- P *says* X: P dice el mensaje X (en la actual ejecución del protocolo).
- P *has* X: X se encuentra disponible para P desde un inicio, P recibió X o P generó X recientemente.
- P *controls* X: El participante P tiene jurisdicción sobre el mensaje X.
- $fresh(X)$: X no ha sido enviado en ningún mensaje anterior al mensaje actual que lo contiene.
- $P \overset{K}{\longleftrightarrow} Q$: Existe una llave K (simétrica) compartida entre P y Q para la comunicación entre ambas entidades. K sólo debe ser conocida por P y por Q y por alguna otra entidad en quien ellos confíen.
- $PK(P,K)$: K es la llave pública del participante P. K^{-1} es la correspondiente llave privada de P (el término -1 no indica potencia, solo se trata de notación). En SVO se hace una clasificación de las llaves públicas de acuerdo a su uso; en $PK_{\psi}(P,K)$ K es la llave pública para cifrado; en $PK_{\sigma}(P,K)$ K es la llave pública para firmas digitales; y en $PK_{\delta}(P,K)$ K es la llave pública para acuerdo de llaves.
- $SV(X,K,Y)$: Verificación de firma. Indica que, dado el mensaje firmado X, aplicando la llave K a este como una llave de verificación de firma se verifica Y como el mensaje firmado con la correspondiente llave privada.
- $\{X\}_K$: Representa una transformación del mensaje X utilizando la llave K. Dicha transformación puede ser una operación de cifrado (simétrico o asimétrico) o la aplicación de una función Hash al mensaje X junto con la llave K. Para el caso de esquemas de cifrado asimétrico, la llave K presenta una llave complementaria denominada $\tilde{K} = K^{-1}$, que constituye la llave privada. En los esquemas de llave simétrico, dado que sólo existe una llave compartida, $\tilde{K} = K$.

- $[X]_K$: Representa el mensaje X firmado digitalmente con la llave K.
- $\langle X \rangle_{*P}$: X de acuerdo a P. Se utiliza para expresar mensajes que P no conoce o reconoce. Un ejemplo podría se un reto aleatorio que no haya sido generado por P o un mensaje $\{X\}_K$ en donde P no conozca la llave K.
- $X\ from\ P$: P envió X.

La representación de los mensajes de los protocolos dentro de SVO se hace empleando una notación muy similar a la notación presentada en el capítulo uno, por lo que se utilizará dicha notación para la escritura de los protocolos criptográficos expuestos en este capítulo.

2.1.3. Reglas de inferencia en SVO

SVO cuenta con dos reglas de inferencia [28]:

- *Modus Ponens* (MP): De φ y $\varphi \supset \psi$ se infiere ψ.
- *Necessitation* (N): De $\vdash \varphi$ se infiere $\vdash P\ believes\ \varphi$

La regla "*Modus Ponens*" es una de las reglas de inferencia básicas (y principal) heredada de la lógica de proposiciones. En latín significa "la forma en que se afirma afirmando" y es utilizada para para la eliminación de implicaciones. Replanteada de una manera sencilla, "Modus Ponens" indica que si de un antecedente (φ) se tiene un consecuente (ψ), si el antecedente ocurre (o es verdadero), entonces se puede afirmar el consecuente.

La regla "*Necessitation*" es la regla más importante de la lógica modal y básicamente indica que si una fórmula es un teorema, entonces cualquier fórmula que se forme a partir de un operador modal aplicado a dicho teorema también es considerado como un teorema. "$\vdash$" es un símbolo metalingüístico que indica que una fórmula φ es derivable sólo a partir de axiomas, por lo que esa fórmula es un teorema. Es importante mencionar que esta regla sólo puede ser aplicada a teoremas (o axiomas), por lo que no se debe utilizar en las premisas formuladas para un análisis particular.

2.1.4. Axiomas de SVO

La lógica SVO cuenta con veinte axiomas, los cuales se clasifican de acuerdo a propósitos comunes. Se tienen entonces axiomas de creencia, de asociación de fuente, de acuerdo de llaves, de recibimiento, de posesión, de comprensión, de decir, de jurisdicción, de frescura, de verificación de *nonce* y de calidad de las llaves simétricas compartidas.

Axiomas de Creencia

Ax1. $P\ believes\ \varphi\ \wedge\ P\ believes\ (\varphi \supset \psi)\ \supset\ P\ believes\ \psi$

Ax2. $P\ believes\ \varphi\ \supset\ P\ believes\ (P\ believes\ \psi)$

El axioma uno indica que un participante P cree todo lo que lógicamente se siga a sus creencias. En otras palabras, si un participante P cree que de un antecedente se sigue un consecuente, y cree además en el antecedente, entonces se puede afirmar que también cree en el consecuente. El axioma dos dice que en efecto un participante puede decir (o afirmar) en lo que el cree.

Axiomas de Asociación de Fuente

Ax3. $(P \overset{K}{\longleftrightarrow} Q\ \wedge\ R\ received\ \{X\ from\ Q\}_K)\ \supset\ (Q\ said\ X\ \wedge\ Q\ has\ K)$

Ax4. $(PK_\sigma(Q,K)\ \wedge R\ received\ X\ \wedge\ SV(X,K,Y))\ \supset\ Q\ said\ Y$

En el axioma tres se indica que si existe un secreto compartido (llave simétrica) entre la entidad P y la entidad Q, y se recibe un mensaje X cifrado con esa llave proveniente de Q, se sigue que es posible distinguir que Q fue quien emitió dicho mensaje. Se asume que los participantes honestos pueden indicar de forma correcta el origen del mensaje, y se asume también que las entidades que poseen la llave K pueden evaluar de forma correcta el origen del mensaje.

En el axioma cuatro es la contra parte al axioma tres para esquemas de cifrado asimétrico. Si se conoce que K es la llave para verificación de firma de Q, y se recibe un mensaje X, y se verifica que X es el resultado de firmar el mensaje Y con la llave complementaria de K (en este caso la llave privada de Q), entonces se infiere que Q dijo el mensaje Y.

Axiomas de Acuerdo de Llaves

Ax5. $((PK_\delta(P,K_p))\ \wedge\ (PK_\delta(Q,K_q)))\ \supset\ P \overset{F_0(K_p,K_q)}{\longleftrightarrow} Q$

Ax6. $\varphi \equiv \varphi[F_0(K,K')/F_0(K',K)]$

Estos axiomas son los que caracterizan el acuerdo de llaves Diffie-Hellman. En el axioma cinco se indica que si existe una llave K_p del participante P y una llave K_q del participante Q ambas para el acuerdo de llaves, entonces se sigue que existe una llave buena (fuerte para fines criptográficos) que es función de estas dos llaves y que solo es compartida por las entidades P y Q. El axioma seis menciona que dos fórmulas son lógicamente equivalentes sin importar el orden de aplicación de las llaves dentro de un acuerdo de llaves.

Axiomas de Recepción

Ax7. $P\ received\ (X_1, ..., X_n)\ \supset\ P\ received\ X_i,\ para\ i = 1, ..., n$

Ax8. $(P\ received\ \{X\}_K\ \wedge\ P\ has\ \tilde{K})\ \supset\ P\ received\ X$

Ax9. $P\ received\ [X]_K\ \supset\ P\ received\ X$

El axioma siete dice que si un participante P recibe un mensaje que esta formado por la concatenación de varios mensajes, entonces P recibe cada uno de los mensajes que conforman el mensaje concatenado. El axioma ocho indica que si P recibe un mensaje cifrado con la llave K, y P posee la llave complementaria $\tilde{K}$ ($\tilde{K} = K$ en el caso de cifrado simétrico, y $\tilde{K} = K^{-1}$ para el caso de cifrado asimétrico), entonces se sigue que P recibe el mensaje X. En el axioma nueve se indica que si el participante P recibe un mensaje X firmado digitalmente, entonces P recibe el mensaje X. Se asume (por conveniencia) que los participantes se encuentran en posesión de las llaves públicas.

Axiomas de Posesión

Ax10. $P\ received\ X\ \supset\ P\ has\ X$

Ax11. $P\ has\ (X_1, ..., X_n)\ \supset\ P\ has\ X_i,\ para\ i = 1, ..., n$

Ax12. $(P\ has\ X_1\ \wedge\ ...\ \wedge\ P\ has\ X_n)\ \supset\ (P\ has\ F(X_1, ..., X_n))$

Los axiomas diez y once básicamente indican que un participante se encuentra en posesión de todo aquello que recibe. En el axioma doce, F es una meta-notación para cualquier función que sea viable calcular para P, como por ejemplo una operación de cifrado. Entonces, si el participante P posee varios mensajes (los cuales posiblemente forman parte de algún otro mensaje concatenado), se infiere que posee también alguna función de ellos.

Axioma de Comprensión

Ax13. $P\ believes\ (P\ has\ F(X))\ \supset\ P\ believes\ (P\ has\ X)$

La función F es una meta-notación para representar a cualquier función uno a uno tal que sea viable calcular F o F^{-1} (su función inversa). Esta puede ser una función de cifrado o de descifrado. Este axioma indica que si el participante P comprende un mensaje X y se encuentra en posesión de una función del mismo, entonces P entiende que lo que posee es X.

Es importante notar que este axioma no implica que P pueda invertir la función P, es decir, encontrar X a partir de $F(X)$. En realidad, si P puede calcular F pero no invertirla,

entonces el axioma trece indica que P sólo conoce $F(X)$ sí P tiene el conocimiento de que posee X. Esto ocurre cuando por ejemplo F representa a una función Hash.

Axiomas de Decir

Ax14. $P\ said\ (X_1, ..., X_n)\ \supset\ (P\ said\ X_i\ \wedge\ P\ has\ X_i),\ para\ i = 1, ..., n$

Ax15. $P\ says\ (X_1, ..., X_n)\ \supset\ (P\ said\ (X_i, ..., X_n)\ \wedge\ P\ says\ X_i),\ para\ i = 1, ..., n$

El axioma catorce indica que un participante quien alguna vez ha dicho un mensaje formado a partir de la concatenación de otros mensajes, también ha dicho y posee las partes concatenadas del mensaje. Un participante posee todo lo que dice. Con respecto al axioma quince, si un participante recientemente ha dicho un mensaje concatenado, entonces dicho participante lo ha dicho y lo dice. La frase "recientemente ha dicho" hace alusión a que el participante dijo ese mensaje en la actual ejecución del protocolo que se analiza.

Axioma de Jurisdicción

Ax16. $(P\ controls\ \varphi\ \wedge\ P\ says\ \varphi)\ \supset\ \varphi$

Este axioma dice que en efecto la palabra de un participante P es una ley para la fórmula en cuestión, por lo que no se debe cuestionar.

Axiomas de Frescura

Ax17. $fresh(X_i)\ \supset\ fresh(X_1, ..., X_n),\ para\ i = 1, ..., n$

Ax18. $fresh(X_1, ..., X_n)\ \supset\ fresh(F(X_1, ..., X_n))$

El axioma diecisiete indica que si una parte de un mensaje concatenado es reciente, entonces a dicho mensaje concatenado se le considera reciente. El axioma dieciocho dice que si un mensaje formado a partir de la concatenación de varios mensajes es reciente, entonces un mensaje generado a partir de una función F de esos mensajes concatenados también se le considera reciente.

Axioma de Verificación de Nonce

Ax19. $(fresh(X)\ \wedge\ P\ said\ X)\ \supset\ P\ says\ X$

Este axioma indica que se puede promover un mensaje de haber sido dicho (en cualquier tiempo) a ser dicho en la época actual (en la ejecución actual del protocolo).

Axioma de Calidad de las Llaves Simétricas Compartidas

Ax20. $P \overset{K}{\longleftrightarrow} Q \equiv Q \overset{K}{\longleftrightarrow} P$

Este axioma indica que una llave compartida entre dos entidades P y Q es buena (fuerte para fines criptográficos) sí y solo sí también es buena para Q y P.

2.1.5. Análisis de protocolos

El análisis sintáctico en SVO se lleva a cabo en dos pasos principales. Primero, se describen las premisas que reflejan las suposiciones basadas en la descripción del protocolo. Segundo, se prueban las metas deseadas del protocolo utilizando las premisas iniciales junto con los axiomas y las reglas de inferencia. Las premisas que reflejan las suposiciones iniciales pueden agrupadas en cuatro tipos. Estos son:

1. Suposiciones de inicio. Son aquellas que al inicio del protocolo se asumen como ciertas. Todo protocolo parte de ciertas suposiciones, como por ejemplo el conocimiento de una llave o secreto compartido entre dos entidades.

2. Premisas que reflejan la recepción de mensajes enviados en una ejecución del protocolo. Constituyen la adaptación de los mensajes del protocolo a fórmulas. Se obtienen directamente de la especificación del protocolo.

3. Premisas que reflejan lo que cada participante comprende de los mensajes que recibe. Aunque un participante P reciba un mensaje X, esto no implica que dicho participante puede comprender todo el contenido del mismo.

4. Premisas que reflejan la interpretación que un participante hace a un mensaje que recibe. Indican lo que un participante receptor interpreta que quiso decir el participante que envió el mensaje.

Como se mencionó, en el segundo paso del análisis se busca probar que las metas de un protocolo de autenticación se cumplan. Algunas de las metas que genéricas que los protocolos de autenticación deben cumplir son las siguientes [29]:

G1. Autenticación Ping: P *believes* Q *says* X

G2. Autenticación de Entidad: P *believes* Q *says* $F(X, N_p)$

G3. Establecimiento Seguro de Llave: P *believes* $P \overset{K-}{\longleftrightarrow} Q$
donde $P \overset{K-}{\longleftrightarrow} Q \equiv (P \overset{K}{\longleftrightarrow} Q \ \wedge \ P \ has \ K)$

G4. Confirmación de Llave: $P\ believes\ (P \overset{K-}{\longleftrightarrow} Q\ \wedge\ Q\ says\ F(K))$

G5. Llave Reciente: $P\ believes\ fresh(K)$

G6. Entendimiento Mutuo de Llaves Compartidas: $P\ believes\ Q\ says\ (Q \overset{K-}{\longleftrightarrow} P)$

La meta **G1** captura situaciones en donde un participante P desea conocer si el otro participante Q se encuentra activo en el presente. El participante Q envía información, la cual P puede comprobar que es reciente (en el tiempo actual de la ejecución del protocolo). En la meta **G2**, N_p es un *nonce* perteneciente a P, y F es una función uno a uno tal que Q puede calcularla en la práctica. La idea de esta meta es que P pueda asegurarse que Q ha dado una respuesta reciente al reto N_p. La meta **G1** constituye un caso particular de **G2**.

La meta **G3** indica que un participante P posee y conoce una llave para comunicarse con Q. Se puede notar que es posible que esta meta se cumpla aún si Q no ha participado en el protocolo y aún si Q no tiene conocimiento de la llave K, por lo que se dice que es un secreto no confirmado. La meta **G4** tiene la intención de describir escenarios en donde un participante P cree que un interlocutor Q ha probado el haber recibido con éxito la llave para la comunicación entre ambos. La forma en que Q demuestra haber recibido el secreto puede variar de protocolo en protocolo, pero básicamente consiste en utilizar esa llave en alguno de los mensajes de la ejecución actual del protocolo.

La meta **G5** simplemente requiere que el participante P crea o tenga la forma de validar que la llave establecida se generó a partir de la ejecución del protocolo en cuestión. Finalmente la meta **G6** se aplica a situaciones donde un participante P puede establecer que un interlocutor Q ha enviado una llave K como un secreto no confirmado entre ellos (desde el punto de vista de Q). Cuando todas estas metas se cumplen para ambos participantes, se puede decir que un protocolo de autenticación es seguro.

A continuación se procede al análisis de dos protocolos de autenticación mediante el uso de la lógica SVO.

2.1.5.1. Análisis del protocolo de Needham y Schroeder

Como se describió en el capítulo uno, el protocolo de Needham y Schroeder consiste de los siguientes pasos (se ha sustituido KDC por S):

1. $A \rightarrow S: \quad ID_a, ID_b, N_a$
2. $S \rightarrow A: \quad \{N_a, ID_b, K_{a,b}, \{K_{a,b}, ID_a\}_{Kb,s}\}_{Ka,s}$
3. $A \rightarrow B: \quad \{K_{a,b}, ID_a\}_{Kb,s}$
4. $B \rightarrow A: \quad \{N_b\}_{Ka,b}$
5. $A \rightarrow B: \quad \{N_b - 1\}_{Ka,b}$

Suposiciones de inicio

Este es el primer paso en el análisis. Las suposiciones iniciales, como se mencionó en el apartado anterior, son aquellas que se obtienen a partir de la especificación del protocolo. Estas sirven de premisas, las cuales se utilizarán junto con los axiomas y las reglas de la lógica para derivar conclusiones. Es común encontrar dentro de las premisas de inicio suposiciones como las creencias que poseen los participantes acerca de la frescura de los *nonces* que generan, las creencias que tienen acerca de que existe una tercera entidad confiable la cual posee la autoridad para generar llaves de sesión fuertes y recientes, o las creencias que tienen acerca de que comparten una buena llave de cifrado (de largo término) con una tercera entidad confiable. Expresadas de forma formal, se tiene:

P1 $A\ believes\ fresh(N_a)$, $B\ believes\ fresh(N_b)$
P2 $A\ believes\ S\ controls(A \overset{Ka,b}{\longleftrightarrow} B)$, $B\ believes\ S\ controls(A \overset{Ka,b}{\longleftrightarrow} B)$
P3 $A\ believes\ S\ controls(fresh(K_{a,b}))$, $B\ believes\ S\ controls(fresh(K_{a,b}))$
P4 $A\ believes(A \overset{Ka,s}{\longleftrightarrow} B)$, $B\ believes(A \overset{Kb,s}{\longleftrightarrow} B)$
P5 $A\ believes\ A\ has(K_{a,s})$, $B\ believes\ B\ has(K_{b,s})$

Recepción de mensajes

Dentro de estas premisas se asume que cada participante recibe los mensajes que le fueron enviados. Constituyen la transcripción del protocolo al lenguaje formal. Se pueden omitir aquellos mensajes que no aporten algo significativo al análisis, como por ejemplo aquellos que son enviados en claro (sin cifrar). Los mensajes sin cifrado no constituyen una fuente de análisis seguro debido a que un adversario puede alterarlos sin problemas. Por esta razón, se omite el mensaje uno del protocolo.

P6 $A\ received\ \{N_a, ID_b, K_{a,b}, \{K_{a,b}, ID_a\}_{Kb,s}\}_{Ka,s}$
P7 $B\ received\ \{K_{a,b}, ID_a\}_{Kb,s}$
P8 $A\ received\ \{N_b\}_{Ka,b}$
P9 $B\ received\ \{N_b - 1\}_{Ka,b}$

Comprensión

Los mensajes recibidos no necesariamente son entendidos. En este conjunto de premisas se establece que partes de un mensaje son claras (comprendidas) para una entidad que lo recibe. Una entidad puede comprender partes del mensaje basándose en la redundancia del mismo, aunque existirán partes en las que no pueda conocer o acceder al contenido. Todo

aquel valor aleatorio que no haya sido generado (previamente) por la entidad que recibe el mensaje, se le considera como algo desconocido o no comprensible para dicha entidad.

P10 A *believes* A *received* $\{N_a, ID_b, \langle K_{a,b}\rangle_{*A}, \langle\{K_{a,b}, ID_a\}_{Kb,s}\rangle_{*A}\}_{Ka,s}$
P11 B *believes* B *received* $\{\langle K_{a,b}\rangle_{*B}, ID_a\}_{Kb,s}$
P12 A *believes* A *received* $\{\langle N_b\rangle_{*A}\}_{Ka,b}$
P13 B *believes* B *received* $\{N_b - 1\}_{Ka,b}$

Se puede observar que en la premisa nueve (P9), la entidad A no conoce de antemano la llave de sesión generada por S, ya que se trata de un valor aleatorio. A su vez, dentro del mismo mensaje, A no tiene la posibilidad de comprender la parte del mensaje cifrada con la llave de sesión $K_{b,s}$, por lo que sólo supone que es un valor aleatorio que debe enviar a la entidad B.

Interpretación

Finalmente, se incluyen las premisas que corresponden al significado que los participantes le dan a los mensajes que reciben. Este tipo de suposiciones son las más difíciles de realizar, y se deben establecer con mucha precaución. Es conveniente razonarlas con detenimiento antes de escribirlas como premisas.

P14 A *believes*(A *received* $\{N_a, ID_b, \langle K_{a,b}\rangle_{*A}, \langle\{K_{a,b}, ID_a\}_{Kb,s}\rangle_{*A}\}_{Ka,s}$ $\supset$
A *received* $\{N_a, ID_b, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A}), \langle\{K_{a,b}, ID_a\}_{Kb,s}\rangle_{*A}\}_{Ka,s}$)
P15 B *believes*(B *received* $\{\langle K_{a,b}\rangle_{*B}, ID_a\}_{Kb,s}$ $\supset$
B *received* $\{A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*B}), ID_a\}_{Kb,s}$)

En la premisa catorce (P14), se ha sustituido la llave $K_{a,b}$ por las expresiones $A \overset{K_{a,b}}{\longleftrightarrow} B$ y $fresh(K_{a,b})$. Estas expresiones indican la interpretación que el participante A hace acerca de ese valor aleatorio. A interpreta que existe una buena llave para la comunicación con B y que además es reciente. La misma interpretación se aplica para la premisa quince, pero esta vez desde el punto de vista de B.

Derivaciones para el participante A

Las derivaciones del participante A se describen a continuación. Cada derivación es numerada anteponiendo la letra que identifica al participante a un número que especifica el número de derivación. Se identifica a aquella derivación que cumpla con alguna de las metas señaladas al inicio de esta sección mediante la anotación del número de meta que cumple (G1, G2, por ejemplo).

A1 A *believes* A *received* $\{N_a, ID_b, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A}), \langle\{K_{a,b}, ID_a\}_{Kb,s}\rangle_{*A}\}_{Ka,s}$
Por P10, P14, N, Ax1, MP

A2 A *believes* S *said*($N_a, ID_b, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A}), \langle\{K_{a,b}, ID_a\}_{Kb,s}\rangle_{*A}$)
Por P4, A1, N, Ax3, Ax2, MP

A3 A *believes* S *says*($N_a, ID_b, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A}), \langle\{K_{a,b}, ID_a\}_{Kb,s}\rangle_{*A}$)
Por P1, Ax17, Ax19, A2, N, Ax19, Ax1, MP

A4 A *believes*$(A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B)$
Por A3, Ax15, P2, N, Ax16, Ax1, MP

A5 A *believes* A *has*$(\langle K_{a,b}\rangle_{*A})$
Por P10, P5,Ax8, Ax10, Ax11, N, Ax1, MP

A6 A *believes*($A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B \ \wedge \ A$ *has*$(\langle K_{a,b}\rangle_{*A})$) **G3**
Por A4, A5

A7 A *believes* $fresh(\langle K_{a,b}\rangle_{*A})$ **G5**
Por A3, Ax15, P3, N, Ax16, Ax1, MP

Como se puede ver del análisis, no se cumplen todas la metas para el usuario A dentro de este protocolo. Sólo se cumplen las metas **G3** y **G4** las cuales indican que A cuenta con una llave no confirmada por parte de la otra entidad a quien corresponde y que además es reciente. Esto quiere decir que dentro de este protocolo la entidad A no puede saber con certeza si la entidad B se encuentra activa en el momento de la ejecución.

Un punto importante a reflexionar a partir de este análisis es que aunque la entidad B sí le responde a A (mensaje cuatro) para anunciarle que se encuentra presente, A no tiene manera de determinar que B realmente dijo ese mensaje. Esto se debe a que en su respuesta B no proporciona algún indicio que sugiera que se trata de él (por ejemplo, un identificador). El *nonce* aleatorio N_b cifrado con la llave de sesión $K_{a,b}$ no es garantía de que B lo haya enviado ya que la entidad A desconoce por completo su valor. Si un adversario modifica el mensaje cuatro (aunque sea por un solo bit), A no tendrá la forma de detectarlo y sólo se limitará a descifrar este mensaje con $K_{a,b}$, decrementar en uno lo que obtenga, volver a cifrar con $K_{a,b}$ (para formar el mensaje cinco) y finalmente enviar una respuesta a B creyendo que ha completado el proceso.

Derivaciones para el participante B

Las derivaciones para el participante B son las siguientes:

B1 B *believes* B *received* $\{A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*B}), ID_a\}_{Kb,s}$
Por P11, P15, Ax1, MP

B2 B *believes* S *said* ($A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*B}), ID_a$)
Por P4, B1, N, Ax3, Ax1, MP
B3 B *believes* S *said* ($A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B$)
Por B2, N, Ax14, Ax1, MP
B4 B *believes* S *said* $fresh(\langle K_{a,b}\rangle_{*B})$
Por B2, N, Ax14, Ax1, MP

De acuerdo a las deducciones obtenidas a partir del análisis, se puede observar que para B no se cumplen ninguno de los objetivos formales preestablecidos. El principal problema que se presenta para no poder terminar el análisis es que el participante B no tiene garantías para validar que la llave recibida en el mensaje tres es reciente (generada para la actual ejecución del protocolo). De esta manera, B no puede dar por hecho que la llave recién adquirida es una buena llave para comunicarse con la entidad A, ya que, como se ha mencionado en la literatura [15][20][29], esta llave podría haberse utilizado con anterioridad y estar comprometida.

2.1.5.2. Análisis del protocolo de Station-to-Station (STS)

El protocolo STS, como se expuso en el capítulo uno, es un protocolo que se basa en el intercambio de llaves de Diffie-Hellman y consiste de los siguientes pasos:

1. $A \rightarrow B:$ Y_a
2. $B \rightarrow A:$ $Y_b, (ID_b, K_b, [ID_b, K_b]_{K_t^{-1}}), \{[Y_b, Y_a]_{K_b^{-1}}\}_{K_{a,b}}$
3. $A \rightarrow B:$ $Y_a, (ID_a, K_a, [ID_a, K_a]_{K_t^{-1}}), \{[Y_a, Y_b]_{K_a^{-1}}\}_{K_{a,b}}$

Donde $(ID_A, K_a, [ID_A, K_a]_{K_t^{-1}}) = Cert_A$ y $(ID_B, K_b, [ID_B, K_b]_{K_t^{-1}}) = Cert_B$ y T es la autoridad certificadora que posee la llave privada K_t para firmas.

Suposiciones de inicio

Las suposiciones iniciales para el protocolo STS son las siguientes:

P1 A *believes* $PK_\sigma(T, K_t)$, B *believes* $PK_\sigma(T, K_t)$
P2 A *believes* $SV(\ [ID_b, K_b]_{Kt^{-1}}, K_t, (ID_b, K_b)\ )$
A *believes* $SV(\ [\langle Y_b\rangle_{*A}, Y_a]_{Kb^{-1}}, K_b, (\langle Y_b\rangle_{*A}, Y_a)\ )$
B *believes* $SV(\ [ID_a, K_a]_{Kt^{-1}}, K_t, (ID_a, K_a)\ )$
B *believes* $SV(\ [\langle Y_a\rangle_{*B}, Y_b]_{Ka^{-1}}, K_a, (\langle Y_a\rangle_{*B}, Y_b)\ )$
P3 A *believes* $fresh(Y_a)$, B *believes* $fresh(Y_b)$
P4 A *believes* $PK_\delta(A, Y_a)$, B *believes* $PK_\delta(B, Y_b)$
P5 A *believes* A *has*(Y_a, X_a), B *believes* B *has*(Y_b, X_b)

P6 A *believes*(T *said* $PK_\sigma(B, K_b) \supset PK_\sigma(B, K_b)$)
B *believes*(T *said* $PK_\sigma(A, K_a) \supset PK_\sigma(A, K_a)$)

La premisa uno indica que tanto el participante A como el participante B confían en K_t como la llave de firmas de la entidad certificadora T. La premisa dos indica que A y B pueden verficar cualquier firma que provenga de T o de alguna otra entidad avalada por T (como A y B). Las premisa tres y cuatro hacen referencia a que A y B creen en que han generado una cantidad aleatoria reciente (Y_a, Y_b) la cual tiene la función de ser una llave para el acuerdo de llaves Diffie-Hellman. La premisa cinco hace alusión al hecho de que si A y B generaron las cantidades aleatorias X_a y Y_a, entonces se encuentran en posesión de las mismas. En la preposición seis se indica que efectivamente T tiene jurisdicción sobre las llaves para firmas. Nótese que no es posible utilizar la expresión A *believes* T *controls* $PK_\sigma(B, K_b)$ debido a que la semántica para la expresión "*controls*" sólo otorga jurisdicción en las sentencias establecidas por T durante la ejecución actual del protocolo.

Recepción de mensajes

Las premisas de recibimiento se presentan a continuación.

P7 B *received* (Y_a)
P8 A *received* $(Y_b, (ID_b, K_b, [ID_b, K_b]_{K_t^{-1}}), \{[Y_b, Y_a]_{K_b^{-1}}\}_{K_{a,b}})$
P9 B *received* $(Y_a, (ID_a, K_a, [ID_a, K_a]_{K_t^{-1}}), \{[Y_a, Y_b]_{K_a^{-1}}\}_{K_{a,b}})$

Comprensión

Para las premisas de comprensión, las únicas expresiones no entendibles en un inicio son los valores aleatorios recién generados para llevar a cabo el acuerdo de llaves. Las premisas de comprensión son las siguientes:

P10 B *believes* B *received* $(\langle Y_a \rangle_{*B})$
P11 A *believes* A *received* $(\langle Y_b \rangle_{*A}, (ID_b, K_b, [ID_b, K_b]_{K_t^{-1}}), \{[\langle Y_b \rangle_{*A}, Y_a]_{K_b^{-1}}\}_{K_{a,b}})$
P12 B *believes* B *received* $(\langle Y_a \rangle_{*B}, (ID_a, K_a, [ID_a, K_a]_{K_t^{-1}}), \{[\langle Y_a \rangle_{*B}, Y_b]_{K_a^{-1}}\}_{K_{a,b}})$

Interpretación

Algunas de las interpretaciones por parte de los participantes son las siguientes:

P13 A *believes*((A *received* $\{[\langle Y_b \rangle_{*A}, Y_a]_{K_b^{-1}}\}_{K_{a,b}} \wedge PK_\sigma(B, K_b) \wedge$
$PK_\delta(A, Y_a)) \supset PK_\delta(B, \langle Y_b \rangle_{*A})$)

P14 B *believes*((B *received* $\{[\langle Y_a \rangle_{*B}, Y_b]_{K_a^{-1}}\}_{K_{a,b}}$ $\wedge$ $PK_\sigma(A, K_a) \wedge$
$PK_\delta(B, Y_b)$) $\supset$ $PK_\delta(A, \langle Y_a \rangle_{*B})$)
P15 A *believes*(T *said*(ID_b, K_b) $\supset$ T *said* $PK_\sigma(B, K_b)$)
B *believes*(T *said*(ID_a, K_a) $\supset$ T *said* $PK_\sigma(A, K_a)$)

En la premisa trece, el participante A entiende que si recibe y confía en los elementos necesarios para creer en la firma de B, entonces podrá confiar en que el valor Y_b fue generado por B para cumplir la función de llave para el acuedo de llaves en la ejecución actual del protocolo. La premisa catorce es similar a la trece pero aplica desde el punto de vista de B. La premisa quince indica que si la entidad certificadora T dijo que hay una llave pública K_b para B, entonces esa llave corresponde a una llave que se utiliza para firmas digitales por parte de B.

Derivaciones para el participante A

A continuación se presentan las derivaciones para el participante A:

A1 A *believes* A *received* $[ID_b, K_b]_{K_t^{-1}}$
Por P11, Ax7, N, Ax1, MP
A2 A *believes* T *said* $PK_\sigma(B, K_b)$
Por A1, P1,P2, P15, N, Ax1, MP
A3 A *believes* $PK_\sigma(B, K_b)$
Por A2, P6, Ax1, MP
A4 A *believes* A *received* $\{[\langle Y_b \rangle_{*A}, Y_a]_{K_b^{-1}}\}_{K_{a,b}}$
Por P11, Ax7, N, Ax1, MP
A5 A *believes* $PK_\delta(B, \langle Y_b \rangle_{*A})$
Por A3, A4, P4, P13, Ax1, MP
A6 A *believes* $(A \overset{K_{a,b}}{\longleftrightarrow} B)$, $K_{a,b} = F_0(Y_a, Y_b)$
Por P4, A5, Ax5, N, Ax1, MP
A7 A *believes* A *has*$(K_{a,b})$
Por P5, P11, Ax10, Ax11, Ax12, N, Ax1, MP
A8 A *believes* $(A \overset{K_{a,b}-}{\longleftrightarrow} B) \equiv A$ *believes*($A \overset{K_{a,b}}{\longleftrightarrow} B$ $\wedge$ A *has*$(K_{a,b})$) **G3**
Por A6, A7, Ax1, MP
A9 A *believes* $fresh(K_{a,b})$ **G5**
Por P3, Ax18, N, Ax1, MP
A10 A *believes* A *received* $[\langle Y_b \rangle_{*A}, Y_a]_{K_b^{-1}}$
Por A4, A7, Ax8, N, Ax1, MP

A11 *A believes B said* $(\langle Y_b \rangle_{*A}, Y_a)$
Por A3, A10, P2, Ax4, N, Ax1, MP
A12 *A believes B says* $(\langle Y_b \rangle_{*A}, Y_a)$ **G1, G2**
Por P3, Ax17, A11, Ax19, N, Ax1, MP
A13 *A believes*($A \overset{K_{a,b}}{\longleftrightarrow} B \ \wedge$ *B says* $\{[\langle Y_b \rangle_{*A}, Y_a]_{K_b^{-1}}\}_{K_{a,b}}$) **G4**
Por A8, A12, Ax1, MP

La meta **G1** constituye un caso especial de la meta **G2**, la cual se deriva en A12. Se puede notar que en esta derivación, el valor aleatorio Y_a además de hacer la función de llave para el acuerdo de llaves, también funciona como un reto el cual es respondido por B. En A8 y A9 se demuestran las metas **G3** y **G5** respectivamente. La meta **G4** se deriva en A13. La derivación de esta meta se realizó de forma diferente a lo presentado por Syverson y Oorschot [28]. En el presente trabajo se obtuvo a partir del plantemiento expuesto en [29] para la meta **G4** (descrita al inicio de esta sección) en lugar de utilizar la formulación de **G4** en [28]. Nótese que en esta derivación (A13), la firma de B ($[\langle Y_b \rangle_{*A}, Y_a]_{K_b^{-1}}$) se encuentra cifrada con la llave $K_{a,b}$, operación que corresponde a una función de la llave ($F(K)$) recién generada.

Derivaciones para el participante B

Las derivaciones del participante B para este protocolo son similares a las establecidas para el participante A. Esto se debe a la simetría que presenta el protocolo en el mensaje dos y tres. Por esta razón no se presentarán la derivaciones de B. Una diferencia que se puede hacer notar en las deducciones consiste en el hecho de que la meta **G6** no puede ser derivada para el participante A, pero sí para B. Esto se debe a que en el momento en que A recibe el mensaje dos desconoce la llave $K_{a,b}$, por lo que B no puede expresar que existe una buena llave para la comunicación entre ambos. Por el contrario, cuando B recibe el mensaje tres, en ese momento él ya cuenta con el conocimiento de la llave $K_{a,b}$, por lo que A le confirma a B que dicha llave es una buena llave para comunicarse el uno con el otro. El hecho de que la meta **G6** no se pueda cumplir para el participante A no implica que el protocolo no es seguro. Esta meta se puede considerar secundaria cuando se cumple la meta **G4**.

Como se puede observar, se cumplen las metas requeridas en ambos participantes para garantizar que este protocolo es seguro. Se puede considerar el protocolo STS es un buen protocolo que garantiza la autenticación confiable. Existen otros factores que se pueden considerar a la hora de plantear aplicabilidad para un protocolo en específico. Debido a que la lógica SVO se encarga de validar un protocolo en el nivel abstracto, factores como el procesamiento pasan desapercibidos para esta. Una desventaja que se puede argumentar

acerca de este protocolo es que realiza muchas operaciones computacionales de alto costo. Se llevan a cabo varias operaciones de cifrado asimétrico que podrían repercutir en el desempeño de las entidades participantes, sobre todo si cuentan con recursos muy limitados. Por ejemplo, un adversario podría montar un ataque de denegación de servicio realizando varias peticiones casi simultáneas (o simultáneas) a B, ya que como se ve en la especificación del protocolo, B realiza cálculos (generación de valores aleatorios, exponenciación y cifrado) sin realizar validaciones una vez que recibe el mensaje uno.

2.2. Prácticas prudentes para el diseño de protocolos criptográficos

En esta sección se explicará el trabajo presentando por Martín Abadi y Roger Needham en [30]. En dicho trabajo, se exponen algunas ideas esenciales para el diseño de protocolos de autenticación en forma de principios. Aunque estos principios constituyen una guía informal de diseño, pueden ser utilizados como complemento a los métodos formales de análisis de protocolos ya que proporcionan una idea clara de lo que se debe considerar o no en los protocolos de autenticación. Con ello, se busca evitar muchos de las confusiones y errores típicos cometidos en el diseño de los mismos. Finalmente, se harán algunos comentarios relacionados a situaciones en donde podría no ser conveniente el uso de estos principios.

2.2.1. Principios de diseño de los protocolos de autenticación

Abadi y Needham propusieron once principios, los cuales cubren diversos aspectos a considerar a la hora de diseñar protocolos de autenticación. Esto no implica que todas estas consideraciones se deban cubrir en todos los protocolos, pero proveen de una lista de cosas a las que vale la pena prestar atención para no cometer errores en el diseño y comprometer la seguridad.

Comunicación explícita

Principio 1. Cada mensaje debería decir lo que significa: la interpretación de un mensaje debería depender sólo de su contenido. Con esto, debería ser posible una frase directa (en español, inglés, etc.) describiendo el contenido del mensaje.

Un ejemplo de esto sería la siguiente frase: "Después de recibir un patrón de bits P, S envía a A una llave de sesión $K_{a,b}$ la cual es una buena llave reciente para la comunicación con B" ($\{N_a, ID_b, K_{a,b}\}_{Ka,s}$). Todos los elementos del significado de este mensaje deberían

estar explícitamente representados en el mensaje, de tal manera que alguien que lo recibe pueda entender el significado del mensaje sin algún otro contexto. Por lo tanto, un protocolo de autenticación debería ser lo suficientemente robusto en el sentido de que el significado de cualesquiera de sus mensajes se pueda inferir a partir de sólo su contenido.

Acciones apropiadas

Principio 2. Las condiciones en las que los mensajes actúan deberían ser claramente establecidas de tal manera que cualquier persona que revise un diseño pueda decir si son aceptables o no.

Esto indica que además de ser explícito, el protocolo deberá cumplir con alguna variedad de condiciones. Estas condiciones a menudo consisten de sentencias de verdad que posiblemente se consideran informales, pero que se debe tener cuidado con ellas. Las sentencias de verdad deben de ser lo suficientemente claras de tal forma que no se presten a malas interpretaciones y sean consideradas inapropiadas. Por ejemplo, si alguien cree que la elección de la llave de sesión debería ser establecida por una tercera entidad (de confianza) en lugar de las entidades participantes en una sesión, entonces aquellos protocolos en donde una o ambas entidades eligen la llave no serán aplicables para la persona que mantenga esta política.

Nombramiento

Principio 3. Si la identidad de un participante es esencial para el significado del mensaje, es prudente mencionar explícitamente el nombre del participante en el mensaje.

Los nombres relevantes para un mensaje algunas veces pueden ser deducidos de otros datos y de las llaves de cifrado que se han aplicado. Sin embargo, cuando esta información no se puede deducir, omitirla constituye un error que puede traer serias consecuencias. Como ejemplo, se tiene el protocolo de Woo y Lam [18], el cual es un protocolo de llave simétrica y se presenta a continuación.

1. $A \rightarrow B: \quad ID_a$
2. $B \rightarrow A: \quad N_b$
3. $A \rightarrow B: \quad \{N_b\}_{Ka,s}$
4. $B \rightarrow S: \quad \{ID_a, \{N_b\}_{Ka,s}\}_{Kb,s}$
5. $S \rightarrow B: \quad \{N_b\}_{Kb,s}$

Como se ve, este protocolo no lleva a cabo un establecimiento de llave de sesión, sólo busca que el participante A demuestre su presencia ante B mediante un intercambio de reto-respuesta. A envía su identidad a B (1), entonces B responde con un reto para A (2), A

lo cifra con la llave que comparte con una tercera entidad S y lo devuelve (3), B pasa la respuesta de A a S (4), S extrae el reto de B, lo cifra con la llave que comparte con él, y se lo devuelve (5). Finalmente B verifica que el nonce que envió en un inicio sea el mismo que recibe.

Este protocolo presenta un fallo ya que la conexión entre los mensajes no es suficiente. Siendo concisos, nada conecta la consulta de B a S con la respuesta de S, por lo que se puede realizar el siguiente ataque. Supóngase que B desea establecer contacto con A y C aproximadamente al mismo tiempo. A puede encontrarse fuera de línea y C puede hacerse pasar por A:

1. $C \rightarrow B: \quad ID_a$
1'. $C \rightarrow B: \quad ID_c$
2. $B \rightarrow A: \quad N_b$
2'. $B \rightarrow C: \quad N_b'$
3. $C \rightarrow B: \quad \{N_b\}_{Kc,s}$
3'. $C \rightarrow B: \quad \{N_b\}_{Kc,s}$
4. $B \rightarrow S: \quad \{ID_a, \{N_b\}_{Kc,s}\}_{Kb,s}$
4'. $B \rightarrow S: \quad \{ID_c, \{N_b\}_{Kc,s}\}_{Kb,s}$
5. $S \rightarrow B: \quad \{N_b''\}_{Kb,s}$
5'. $S \rightarrow B: \quad \{N_b\}_{Kb,s}$

Aquí N_b'' es el resultado de descifrar $\{N_b\}_{Kc,s}$ con la llave $K_{a,s}$. En los mensajes 1 y 1', C le dice a B que A y C desean establecer una conexión. En los mensajes 2 y 2', B responde con dos retos; C recibe uno de manera normal y captura el que estaba destinado para A. En los mensajes 3 y 3', C responde a ambos retos. En favor de A, C puede enviar cualquier cosa. En su propio beneficio, C responde con el reto dirigido a A. En los mensajes 4 y 4', B consulta a S acerca de las dos respuestas. Los mensajes 5 y 5' son respuestas de S. Una de estas respuestas no coincide con nada, mientras que la otra contiene el reto destinado para A. Basándose en estas respuestas, B podría creer que A se encuentra presente si el adversario intercambiara los mensajes 5 y 5'.

La existencia de este ataque demuestra que los mensajes en el protocolo no son lo suficientemente explícitos con respecto a la identidad de los participantes en cuestión. Una forma de eliminar este fallo es cambiando el mensaje 5 por el siguiente:

5. $S \rightarrow B: \quad \{ID_a, N_b\}_{Kb,s}$

Con ello se logra la asociación entre los mensajes 4 y 5.

Usos del cifrado

Principio 4. Ser claros en porqué se esta realizando cifrado. Las operaciones de cifrado no son de bajo costo (computacional), y no preguntarse el porqué se están realizando puede conducir a redundancia. Cifrado no es sinónimo de seguridad, y su uso inadecuado puede conducir a errores.

Los usos que se le da al cifrado van desde preservar confidencialidad, garantizar autenticidad y generar números aleatorios hasta enlazar partes de los mensajes con el fin de lograr asociaciones. Es importante saber con exactitud a cual de estos usos se esta recurriendo para poder evitar errores y mal gasto de operaciones. Para ilustrar este principio, a continuación se expone una versión simplificada del protocolo Kerberos versión 4.

1. $A \rightarrow S: \quad ID_a, ID_b$
2. $S \rightarrow A: \quad \{T_s, L, K_{a,b}, ID_b, \{T_s, L, K_{a,b}, ID_a\}_{Kb,s}\}_{Ka,s}$
3. $A \rightarrow B: \quad \{T_s, L, K_{a,b}, ID_a\}_{Kb,s}, \{ID_a, T_a\}_{Ka,b}$
4. $B \rightarrow A: \quad \{T_a + 1\}_{Ka,b}$

En este protocolo, T_s y T_a son estampas de tiempo, y L indica un tiempo de vida. De inicio el servidor S comparte las llaves $K_{a,s}$ y $K_{b,s}$ con A y B respectivamente. Algunos detalles importantes a mencionar son:

- En el mensaje dos se requiere de cifrado. La llave $K_{a,b}$ debe permanecer confidencial, y S la firma como prueba de autenticidad. Por otra parte, en este mismo mensaje, no es necesario que los datos cifrados con la llave $K_{b,s}$ se vuelvan a cifrar junto con otro conjunto de datos, ya que se encuentran protegidos de antemano. Realizar esta operación de doble cifrado, genera mas costo computacional innecesario.

- En la segunda parte del mensaje tres el cifrado se utiliza con una finalidad completamente diferente: A cifra T_a con la llave $K_{a,b}$ con el propósito de probar que conoce la llave $K_{a,b}$ en un tiempo cercano a T_a.

Considerando estos detalles (y otros), se estableció la versión 5 de Kerberos, la cual se extrajo de la especificación [31] y se muestra a continuación en forma idealizada:

1. $A \rightarrow S: \quad ID_a, ID_b, N_a, Params_1$
2. $S \rightarrow A: \quad ID_a, \{T_s, L, K_{a,b}, ID_b, N_a\}_{Ka,s}, \{T_s, L, K_{a,b}, ID_a, Params_1\}_{Kb,s}$
3. $A \rightarrow B: \quad \{T_s, L, K_{a,b}, ID_a, Params_1\}_{Kb,s}, \{ID_a, T_a, Params_2\}_{Ka,b}$
4. $B \rightarrow A: \quad \{T_a, Params_2\}$

Donde $Params_1$, $Params_2$ representan parámetros que le añaden mayor funcionalidad a Kerberos (se explicarán con mayor detalles en el capítulo siguiente). Se puede observar que

entre los pequeños cambios realizados destacan la añadidura de parámetros extra dentro del intercambio y la omisión del doble cifrado en el mensaje dos.

Firmando los datos cifrados

Principio 5. Cuando un participante firma un material que ya ha sido cifrado, no se debería inferir que un participante conoce el contenido del mensaje. Por otro lado, resulta apropiado inferir que el participante que firma un mensaje y después lo cifra para privacidad conoce el contenido del mensaje.

Para ilustrar este principio, se muestra a continuación un protocolo de un mensaje que se encuentra especificado en el estándar CCITT X.509 [32]. Este protocolo tiene el único objetivo que un participante A envíe información íntegra a otro participante B, y que B tenga la posibilidad de verificar que A fue quien efectivamente dijo el mensaje (propiedad de no repudio).

1. $A \rightarrow B: \quad ID_a, \{T_a, N_a, ID_b, X_a, \{Y_a\}_{Kb}\}_{K_a^{-1}}$

Aquí, T_a es una estampa de tiempo, N_a es un nonce, y X_a y Y_a representan información de usuario. En la práctica, este protocolo ocupa operaciones hash para reducir la cantidad de información a cifrar (firmar), aunque esto no afecta la falla a la que se desea hacer énfasis. Como se ve en el protocolo, a pesar de que Y_a se transfiere en un mensaje firmado, no existe evidencia en donde se sugiera que el usuario A está al tanto de la información contenida en esta parte privada del mensaje. Un adversario podría, por ejemplo, interceptar el mensaje y remover la firma existente para añadir la suya, copiando adicionalmente (a ciegas) la información cifrada (Y_a) dentro de la firma, y con ello engañar al participante B. Este problema se puede evitar de varias maneras, una de ellas (y la más simple) es firmar la información secreta (Y_a) antes de ser cifrada para privacidad. Otra alternativa sería utilizar el fundamento del principio tres y añadirle a la información cifrada el identificador del participante A, tal como se muestra a continuación.

1. $A \rightarrow B: \quad ID_a, \{T_a, N_a, ID_b, X_a, \{ID_a, Y_a\}_{Kb}\}_{K_a^{-1}}$

En general, se recomienda firmar antes de cifrar, aunque realizarlo de esta manera no es garantía de que el protocolo se encontrará libre de errores. Se deben establecer de manera clara las asociaciones necesarias dentro de la firma para evitar en lo mayor posible incurrir en errores que propicien vulnerabilidades en la estructura de un protocolo criptográfico.

Estampas de tiempo, números de secuencia, y otros *nonces*

Principio 6. Ser claros en cuales son las propiedades que se asumen acerca de los *nonces*. Lo que posiblemente se hace para asegurar sucesión temporal posiblemente no se utiliza para asegurar asociación (y posiblemente la asociación se establece por otros medios).

El uso común que se les da a los *nonces* es a manera de reto. Un usuario participante envía un *nonce* esperando que el participante a quien va dirigido lo devuelva como respuesta en un mensaje posterior pero dentro de la misma ejecución del protocolo. Esto garantiza sucesión temporal y además permite asegurar la validez de los mensajes. Adicionalmente, los *nonces* también se pueden utilizar para asociar a un usuario, haciendo innecesario el uso de su identificador (ID) en alguno de los mensajes subsecuentes. Un ejemplo de esto se presenta en el protocolo de Otway y Rees [33]. Este se describe a continuación.

1. $A \rightarrow B: \quad M, ID_a, ID_b, \{N_a, M, ID_a, ID_b\}_{Ka,s}$
2. $B \rightarrow S: \quad M, ID_a, ID_b, \{N_a, M, ID_a, ID_b\}_{Ka,s}, \{N_b, M, ID_a, ID_b\}_{Kb,s}$
3. $S \rightarrow B: \quad M, \{N_a, K_{a,b}\}_{Ka,s}, \{N_b, K_{a,b}\}_{Kb,s}$
4. $B \rightarrow A: \quad M, \{N_a, K_{a,b}\}_{Ka,s}$

En los mensajes uno y dos se puede observar que el cifrado del *nonce* N_a y N_b establece una referencia segura hacia los participantes A y B. Por lo que cuando el servidor S responde el mensaje tres, no requiere de utilizar los identificadores de A ni de B en su respuesta. En el caso de que los nonces N_a y N_b se encontraran fuera de la información protegida mediante el cifrado, dejarían de ser una referencia segura hacia las entidades A y B ya que cualquier adversario podría modificarlos.

Citando el ejemplo del protocolo de Woo y Lam expuesto junto con el principio tres, se puede notar que el propósito de cifrar en el mensaje cuatro es para enlazar dos partes de un mensaje. Sin embargo, si se analiza con detenimiento el uso de los *nonces*, se puede observar que N_b sólo provee una prueba de frescura, pero no una asociación al nombre de A (ID_a) como se pretende. Con esto, y gracias al cambio establecido en el mensaje cinco en donde se agrega el identificador de A por motivos de seguridad, el cifrado hecho para enlazar ID_a y N_b en el mensaje cuatro resulta innecesario. Esto hace que el protocolo quede de la siguiente manera:

1. $A \rightarrow B: \quad ID_a$
2. $B \rightarrow A: \quad N_b$
3. $A \rightarrow B: \quad \{N_b\}_{Ka,s}$
4. $B \rightarrow S: \quad ID_a, \{N_b\}_{Ka,s}$
5. $S \rightarrow B: \quad \{ID_a, N_b\}_{Kb,s}$

Principio 7. El uso de una cantidad predecible (como el valor de un contador) puede servir para garantizar frescura, a través de un intercambio de reto-respuesta. Pero para que una cantidad previsible pueda ser efectiva, debería ser protegida de tal forma que un intruso no tenga posibilidad de simular un reto y utilizarlo en ejecución posterior del protocolo.

Es común que los protocolos que confían en el uso de relojes sincronizados estén acompañados por algún protocolo de acceso a los servidores que proporcionan el tiempo. Este tipo de protocolos de acceso al tiempo como tal no pueden confiar en el uso de relojes sincronizados, por lo que deben de confiar en otros elementos como lo son los *nonces* aleatorios o los *nonces* previsibles. Empleando *nonces* aleatorios podemos considerar el siguiente protocolo:

1. $A \rightarrow S: \quad ID_a, N_a$
2. $S \rightarrow A: \quad \{T_s, N_a\}_{Ka,s}$

Aquí, T_s es el tiempo actual (solicitado) y N_a es un *nonce* aleatorio utilizado como reto. Una vez que finaliza este intercambio, el participante A acepta a T_s como el tiempo actual sí la respuesta llega en un lapso considerablemente corto para A una vez que emitió el reto.

Este protocolo no funcionaría si N_a fuera una cantidad previsible (como un contador que se incrementa en cada petición). Un adversario C podría hacer que A retrasara su reloj: en una ejecución temprana del protocolo, C realiza una petición para el tiempo actual (en ese momento) utilizando un valor futuro (no utilizado) del *nonce*, almacena la respuesta del servidor, y entonces la envía a A cuando A utilice este valor como reto.

Cuando N_a es una cantidad previsible, entonces esta debe ir protegida:

1. $A \rightarrow S: \quad ID_a, \{N_a\}_{Ka,s}$
2. $S \rightarrow A: \quad \{T_s, \{N_a\}_{Ka,s}\}_{Ka,s}$

Con la protección del *nonce* mediante el cifrado, este ataque ya no se puede realizar. De esto se puede ver la importancia que tiene el proteger las cantidades previsibles que se utilicen para garantizar la frescura de los mensajes.

Principio 8. Si las estampas de tiempo son utilizadas para garantizar frescura mediante la referencia a un tiempo absoluto, entonces la diferencia entre los relojes locales de varias de las máquinas involucradas debería ser mucho menor al tiempo permisible en el que un mensaje es considerado válido. Además, el mecanismo de mantenimiento del tiempo debe formar parte de la base del sistema de cómputo confiable.

Citando como ejemplo el protocolo de Denning y Sacco presentado en el capítulo uno, se tiene que este protocolo se basa en la confianza de relojes sincronizados para otorgar garantías de frescura en los mensajes transmitidos. Como se mencionó, existe un riesgo importante para cuando se desea garantizar que los mensajes son recientes mediante estampas de tiempo,

este se basa en el hecho de que los relojes de cada uno de los participantes involucrados pueden perder la sincronía a causa de un sabotaje (por parte de un adversario) o fallas en el mecanismo de sincronización. Un adversario podría aplicar el ataque de supresión-repetición, en el cual teniendo el conocimiento de que el reloj de un participante que envía un mensaje se encuentra adelantado, lo captura y lo retransmite cuando el reloj del destinatario se encuentra dentro del rango para aceptar el mensaje. Esto puede ocasionar resultados inesperados por parte de quien recibe el mensaje retransmitido.

Qué es frescura: uso contra generación

Principio 9. Una llave que ha sido utilizada recientemente, por ejemplo para cifrar un nonce, puede ser bastante antigua y estar posiblemente comprometida. Usos recientes de la llave no hacen que la llave sea mejor de lo que sería en otro caso.

Citando el ejemplo del protocolo de Needham y Schroeder, expuesto en el capítulo uno, se puede observar que los mensajes cuatro y cinco tienen la intención de distribuir la llave de sesión $K_{a,b}$ a B y además convencerlo de que A se encuentra activo en ese momento. Aunque es posible que la idea original del protocolo incluya además convencer a B que la llave de sesión es reciente (es decir, que fue generada para esta ejecución del protocolo), se puede notar que este objetivo no se cumple ya que no existe algún parámetro o campo que le indique a B que la llave es fresca. De esta manera, un adversario podría capturar el mensaje cuatro y luego replicarlo sin que B tenga manera de verificar que esa llave ha sido utilizada con anterioridad. Peor aún, si un adversario logra comprometer la llave de sesión, entonces se encontrará en disposición de poder establecer contacto con B cuando lo desee.

Entre las alternativas que surgieron para solucionar este problema en el protocolo de Needham y Schroeder están el protocolo de Denning y Sacco y Kerberos. Ambas hacen uso de relojes sincronizados para garantizar que la llave de sesión entregada a B es reciente. Como se describió en el principio anterior, la desventaja que se tiene en estas alternativas es la sincronía de los relojes. La diferencia entre los relojes locales de cada entidad debe de ser mínima para evitar problemas.

Reconociendo mensajes y codificaciones

Principio 10. Si se utiliza una codificación para presentar el significado de un mensaje, entonces debería ser posible decir cuál es la codificación utilizada. En el caso común en donde la codificación es dependiente de un protocolo, debería ser posible deducir que el mensaje pertenece a este protocolo, y de hecho a una ejecución particular del mismo, y conocer su número en el protocolo.

Este principio básicamente indica que es importante que los participantes puedan reconocer los mensajes por lo que son, y que puedan indicar en todo momento cada etapa de cualesquiera protocolo que ejecuten. Existen dos formas posibles en las que los participantes pueden caer en confusión: una es entre un mensaje actual y un mensaje de propósito similar de alguna ejecución previa, y la otra es entre un mensaje actual y un mensaje que pertenezca a cualquier otro paso del protocolo, o a otro protocolo.

En particular, si un mensaje contiene suficientes garantías de tiempo y nombres, entonces la instancia actual de este mensaje no puede ser confundida con una instancia anterior, o una instancia destinada a otro participante. Podría, sin embargo, confundirse con una instancia concurrente para el mismo participante. Los mensajes se deben elaborar de tal forma que cada uno de ellos debe decir lo que significa y de esa forma no existirá razón para preocuparse por este contexto.

Para ilustrar este principio podemos hablar acerca de la confidencialidad (y firma) de los mensajes mediante el uso de mecanismos de cifrado simétrico. Un problema común que se presenta en los esquemas de cifrado convencional se encuentra asociado a la dirección en la que los mensajes se transfieren. En el protocolo de Needham y Schroeder por ejemplo, el participante B envía un reto N_b en el mensaje cuatro y recibe como respuesta $N_b + 1$ en el mensaje cinco.

4. $B \rightarrow A: \quad \{N_b\}_{Ka,b}$
5. $A \rightarrow B: \quad \{N_b + 1\}_{Ka,b}$

Aquí, el propósito de incrementar N_b es el de poder distinguir el reto de la respuesta. Sin este incremento, un adversario podría enviar de regreso el mensaje cuatro a B, quien podría confundirlo con la respuesta de A. El propósito de modificar un nonce es a menudo interpretado de manera errónea. Por ejemplo, si en el mensaje cuatro se agregara el identificador de B ($\{ID_b, N_b\}_{Ka,b}$), no habría necesidad de incrementar en uno N_b. El simple hecho de que A realice la operación de extraer el nonce N_b del mensaje cuatro y lo coloque como único campo dentro del mensaje cinco hace que ambos mensajes sean distintos y no se requiera de alguna otra modificación al mismo.

Confianza

Principio 11. El diseñador del protocolo debería conocer cuales son las relaciones de confianza de las que depende su protocolo, y el porque de la necesidad de la dependencia. Las razones por las cuales las relaciones particulares de confianza son aceptables deberían ser explícitas de tal manera que estarán fundadas en razones de criterio y políticas en lugar de razones lógicas.

Como ejemplo tenemos el protocolo de "La Rana de Boca Grande" que se expuso en el capítulo uno y se repite en esta sección con fines ilustrativos. Este protocolo consiste de sólo dos mensajes, los cuales son:

1. $A \rightarrow S: \quad ID_a, \{T_a, B, K_{a,b}\}_{Ka,s}$
2. $S \rightarrow B: \quad \{T_s, ID_a, K_{a,b}\}_{Kb,s}$

Primero, A envía un mensaje a S en donde incluye una llave de sesión $K_{a,b}$ y una estampa de tiempo T_a. Entonces S verifica T_a y le envía un mensaje a B, junto con su estampa de tiempo T_s. Finalmente, B valida T_s y acepta la llave de sesión $K_{a,b}$ para la comunicación con A.

En este protocolo se puede observar que en quien se confía para generar la llave de sesión $K_{a,b}$ es en la entidad A. Este tipo de confianza es a menudo inaceptable debido a que se deben cumplir ciertos requerimientos de la calidad de las llaves (como la no repetición y la impredecibilidad) que la entidad que se desea obtener un servicio podría omitir. Es mejor contar con una entidad confiable (como un centro de distribución de llaves) que se encargue de realizar esta tarea.

2.2.2. Comentarios acerca de los principios de diseño para los protocolos criptográficos

Los principios de diseño expuestos por Martín Abadi y Roger Needham en [30] son, como se mencionó, buenas prácticas para la eleaboración de protocolos de autenticación. Su fundamento se encuentra en el hecho de que los protocolos deben de ser lo más explícitos posible. Con ello, se busca evitar las confusiones generadas por la falta de información dentro de los mismos. En general, existen situaciones en donde la anonimia de los mensajes en el intercambio constituye un punto fuerte a favor del protocolo. Un ejemplo de esto se presenta en un protocolo conocido como EKE (Encrypted Key Exchange en inglés), el cual es un protocolo de distribución de llaves que funciona como un multiplicador de privacidad. Consiste del siguiente intercambio de mensajes:

1. $A \rightarrow B: \quad ID_a, \{K_a\}_P$
2. $B \rightarrow A: \quad \{\{K_{a,b}\}_{Ka}\}_P$
3. $A \rightarrow B: \quad \{N_a\}_{Ka,b}$
4. $B \rightarrow A: \quad \{N_a, N_b\}_{Ka,b}$
5. $A \rightarrow B: \quad \{N_b\}_{Ka,b}$

En este protocolo, A es la entidad cliente que desea acceder a los servicios de un servidor B. Ambas entidades comparten una contraseña P como secreto la cual puede ser una contraseña

débil y de baja entropía. A cifra su llave pública K_a con la contraseña y se lo envía a B. B genera una llave de sesión $K_{a,b}$, la cifra con la llave pública de A y el resultado lo cifra nuevamente pero ahora con la contraseña P. Entonces, ocurre un acuerdo en el que ambas entidades demuestran conocer la llave de sesión recién generada.

Lo importante a observar en este protocolo es que el contenido de los mensajes no puede ser confirmado cuando son recibidos por las entidades ya que cuando un participante recibe los mensajes no puede enlazar su contenido a valores conocidos sólo hasta que completa el protocolo. En particular, los participantes no pueden decir si los mensajes recibidos poseen la forma correcta para que ellos puedan continuar o no con el siguiente paso. Sólo cuando cada una de las entidades recibe el último mensaje es cuando pueden confirmar que los mensajes precedentes poseen el contenido correcto. Si alguno de los mensajes llegara a contener la redundancia adecuada (un identificador de usuario por ejemplo) para que un participante pueda conocer que es lo que esta recibiendo (o enviando) antes de que termine la ejecución del protocolo, entonces el protocolo sería vulnerable a los ataques de adivinanza de contraseñas.

Este protocolo no solo constituye un contra ejemplo hacia el carácter explícito de los principios expuestos por Abadi y Needham, si no que también representa un ejemplo para otros principios de diseño. Anderson y Needham en [34] realizaron una extensión a los principios proponiendo principalmente lo siguiente:

- Sé cuidadoso cuando firmas o descifras datos de tal manera que un adversario no pueda utilizarte como un oráculo.

Como se puede ver, el protocolo EKE le da un giro a este argumento, ya que en lugar de prevenir el uso de los participantes como oráculos, se asegura que la salida de estos oráculos (participantes) no sea de utilidad para el adversario. En general, a manera de conclusión y de acuerdo a lo observado con el protocolo EKE, podemos decir que se puede omitir el uso de los principios siempre y cuando sea por una buena razón y esto se haga de manera precavida.

Capítulo 3

Evaluación práctica de los protocolos de autenticación Kerberos y RADIUS.

El presente capítulo tiene como finalidad poner en práctica algunos de los conceptos previamente estudiados de autenticación. Para ello, se expondrán dos de los protocolos principales utilizados en la actualidad, junto con su respectiva evaluación de implementación en escenarios de pruebas específicos. Como primer paso, se analizará un protocolo conocido como Kerberos, que es un protocolo de autenticación que pertenece a la capa de aplicación y que fue desarrollado como parte del proyecto Athena en el MIT (*Massachusetts Institute of Technology* por sus siglas en inglés [35]). Como segundo caso se estudiará un marco de trabajo (*framework* en inglés) conocido como RADIUS (*Remote Authentication Dial-In User Service* por sus siglas en inglés), el cual es un estándar IETF [36] empleado para la integración de una gran variedad de protocolos de autenticación que ayudan a poder llevar a cabo el control de acceso de usuarios en las redes de cómputo distribuido.

3.1. El protocolo de autenticación Kerberos

En esta sección se describe el protocolo de autenticación Kerberos. Comenzaremos con una introdución a este protocolo, planteando el panorama junto con el problema que intenta resolver. Se presenta una descripción general de la operación de Kerberos para consecuentemente detallar la versión 5 del mismo. Finalmente, se presenta una prueba de implementación en la que se muestra la operación real del sistema Kerberos.

3.1.1. Introdución

Kerberos es un servicio de autenticación desarrollado como parte del proyecto Athena en el MIT [35]. El modelo de Kerberos se encuentra basado en parte en el protocolo de autenticación de tres entidades de Needham y Schroeder [14] y en las modificaciones sugeridas por Denning y Sacco [15]. Cuenta con dos versiones disponibles en la actualidad, la versión 4 y la versión 5. La versión 4 fue presentada por primera vez a mediados de los años ochenta y existen diversas implementaciones de esta versión cuyo uso ha sido relegado por las implementaciones de la versión 5. La versión 5 corrige algunas de las deficiencias de seguridad de la versión 4 y se ha convertido en un estándar IETF descrito en el RFC 4120 [31].

El escenario en el que opera Kerberos es el siguiente: Supóngase que se se tiene un ambiente distribuido en el cual los usuarios conectados a través de estaciones de trabajo (computadoras, terminales) desean acceder a diversos servicios implementados en servidores que se encuentran distribuidos en toda la red. Una condición deseable es que los servidores tengan la capacidad de permitir el acceso sólo a los usuarios autorizados. Dentro de este ambiente, no se le puede confiar a las estaciones de trabajo la tarea de identificar a sus usuarios para que utilicen los servicios de la red. Los motivos son los siguientes [1]:

1. Un usuario podrá obtener acceso a una estación de trabajo particular y pretender ser algún otro usuario operando esa estación de trabajo.

2. Un usuario podrá alterar la dirección de red de una estación de trabajo de tal manera que las peticiones enviadas desde de la estación de trabajo modificada parezcan provenir de alguna otra estación de trabajo suplantada.

3. Un usuario podrá escuchar y capturar los intercambios de mensajes para posteriormente repetirlos con el objetivo de obtener acceso a un servidor o de alterar la operación del mismo.

En cualquiera de los casos, es posible que un usuario no autorizado pueda obtener acceso a los servicios y datos a los que no tiene autorizado acceder. En lugar de construir un protocolo de autenticación en cada servidor (lo cual generaría una carga adicional a los mismos), Kerberos provee de un servidor de autenticación centralizado cuya función es autenticar a usuarios con servidores y a servidores con usuarios en un ambiente de red abierto y desprotegido.

El protocolo Kerberos al igual que el protocolo de Needham y Schroeder se basa en el uso de criptografía de llave simétrica. Básicamente el intercambio de mensajes en las versiones 4 y 5 es el mismo presentándose ligeras diferencias en la codificación de los mensajes. Debido a que la versión 4 de Kerberos no es recomendable en la actualidad (porque utiliza el algoritmo DES como única opción de cifrado), sólo se explicará con la versión 5. Una referencia en la que

se explican las diferencias entre ambas versiones se encuentra en [37]. Antes de describir la versión 5 del protocolo en detalle, se introduce primero una descripción general de Kerberos.

3.1.2. Descripción general del protocolo Kerberos

Como se mencionó, la versión 5 del protocolo Kerberos tiene la intención de mejorar algunas de las limitaciones de la versión 4. Al igual que la versión 4, esta versión cuenta con los siguientes elementos en juego:

- **Clientes:** Son los procesos (ejecutados dentro de las estaciones de trabajo) que ayudan a los usuarios a obtener un servicio.
- **Servidores:** Son los recursos disponibles en la red. Proporcionan un servicio al usuario. A los servidores también se les conoce como Servidores de Aplicación.
- **KDC:** Es el Centro de Distribución de Llaves. Constituye un servicio de la red que proporciona *tickets* y llaves de sesión. Se dividide en dos partes o etapas, la primera es el Servidor de Autenticación (AS por sus siglas en inglés) y la segunda el Servidor de Otorgamiento de *Ticket* (TGS por sus siglas en inglés). Ambas etapas se encuentran implementadas en el mismo servidor, es decir, en el mismo equipo físico.

Bajo la terminología de Kerberos, tanto una entidad clientes como una entidad servidora son referidas bajo el nombre de *"participantes"* (*principals* en inglés). Esto indica que, para Kerberos, ambos tipos de entidades son clientes (o usuarios) a los que se les proporciona el servicio de autenticación. Cada usuario del sistema Kerberos posee un identificador único denominado *"nombre del participante"* que consta de un nombre de usuario (ID) y un nombre de dominio. El nombre de dominio es conocido como *"reino"* (*realm* en inglés) y es independiente del nombre de dominio DNS (*Domain Name System* por sus siglas en inglés). En este trabajo se utilizará el nombre de *"dominio Kerberos"* en lugar de la palabra *"reino"*.

Antes de iniciar la descripción del protocolo Kerberos, es importante mencionar algunos detalles. En primer lugar, para que los clientes y servidores puedan utilizar este servicio de autenticación, deberán estar previamente registrados en la base de datos del sistema Kerberos. Durante el registro, a cada cliente (o servidor de aplicación) se le proporciona un identificador único y una contraseña. Con la ayuda de la contraseña y un algoritmo de generación de llaves, se genera a cada participante una llave de cifrado simétrica que será la que se registre en la base de datos del servidor Kerberos (junto con el *ID*). Esta llave representa el secreto compartido entre el cliente y el *KDC* y debe ser fuertemente protegida. En segundo lugar, el modelo de Kerberos se basa en el uso de relojes sincronizados, por lo

que todos los dispositivos que ejecuten este protocolo deberán contar con algún mecanismo de obtención del tiempo dentro de la red (o redes).

El modelo de autenticación de Kerberos puede ser dividido en tres etapas, y cada etapa consiste de un intercambio de dos mensajes, los cuales se muestran en la figura 3.1.

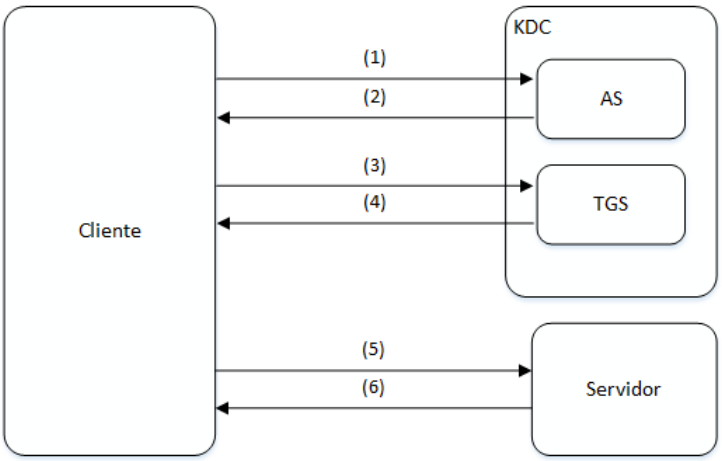

Figura 3.1: Intercambio de mensajes del protocolo Kerberos

En la primera etapa, el cliente (o aplicación cliente del usuario) se comunica con el servidor de autenticación *AS* para obtener credenciales (un ticket y una llave de sesión) que utilizará posteriormente para dirigirse con el servidor de otorgamiento de *ticket TGS*. En la segunda etapa, el cliente establece contacto con el *TGS* presentando el *ticket* otorgado por el *AS*. Con ello, el *TGS* le proporciona al cliente otro *ticket* con credenciales (que incluyen otra llave de sesión) para poder acceder a un servicio específico de la red. Finalmente en la tercera etapa, el cliente se dirige con el servidor del cual desea obtener un servicio, presentando el *ticket* obtenido en la segunda etapa. Validando las credenciales (es decir, autenticando), el servidor decide o no otorgar el servicio al cliente.

Al primer *ticket* obtenido por el cliente se le denomina *"Ticket de otorgamiento de ticket"* y al segundo *"Ticket de otorgamiento de servicio"*. Para evitar que un usuario tenga que ingresar su contraseña cada vez que requiera utilizar un nuevo servicio, Kerberos divide al *KDC* en dos etapas (*AS* y *TGS*) y hace uso de los *tickets* recién mencionados. Así, por ejemplo, cuando un trabajador (usuario) de una compañía accede a su estación de trabajo para iniciar su jornada laboral, este ingresa su nombre de usuario y contraseña; en ese momento la aplicación cliente que maneja el login de inicio podrá comunicarse con el *AS*, y obtener el *Ticket* de otorgamiento de *ticket*. Con este, cada vez que el trabajador requiera acceder a uno o varios de los recursos proporcionados por la red de la empresa (servidor de base de datos, servidor de impresión, etc), recurrirá a presentarlo ante el *TGS* y podrá obtener un *ticket* para cada servicio deseado. Entonces, la aplicación cliente (empleada por el

trabajador) que maneja el acceso a un servicio especifíco, procede finalmente a autenticarse ante el servidor para el cual fue emitido el *ticket*, obteniendo así el acceso al recurso. Todo esto ocurre sin la necesidad de que el trabajador ingrese su contraseña por cada servicio que requiera utilizar.

En otras palabras, dividir al *KDC* en dos etapas tiene como finalidad dos cosas, reducir el riesgo de exposición de la llave secreta del cliente y hacer más transparente para el usuario el uso de Kerberos. Con este modelo, es posible hacer que los usuarios del sistema ni siquiera se den cuenta que están utilizando Kerberos para autenticarse, ya que quienes finalmente manejan y presentan los *tickets* durante las etapas del protocolo son las aplicaciones cliente que los individuos utilizan. Una ventaja adicional que proporciona el uso de *tickets* con credenciales es que estos cuentan con tiempos de vigencia, por lo que si un usuario ha obtenido un *ticket* para un servicio específico, podrá utilizarlo varias veces para autenticarse ante un servidor sin tener que ejecutar las etapas anteriores del protocolo.

3.1.3. Versión 5 del protocolo Kerberos

La codificación de los mensajes de Kerberos V5 se presenta a continuación [1].

1. $C \rightarrow AS: \quad Options, ID_c, Realm_c, ID_{tgs}, Times, N_{c1}$
2. $AS \rightarrow C: \quad Realm_c, ID_c, Ticket_{tgs}, \{K_{c,tgs}, Times, N_{c1}, Realm_{tgs}, ID_{tgs}\}_{K_{c,kdc}}$
   $Ticket_{tgs} = \{Flags, K_{c,tgs}, Realm_c, ID_c, AD_c, Times\}_{K_{tgs,kdc}}$

3. $C \rightarrow TGS: \quad Options, ID_s, Times, N_{c2}, Ticket_{tgs}, Authenticator_c$
4. $TGS \rightarrow C: \quad Realm_c, ID_c, Ticket_s, \{K_{c,s}, Times, N_{c2}, Realm_s, ID_s\}_{K_{c,tgs}}$
   $Ticket_s = \{Flags, K_{c,s}, Realm_c, ID_c, AD_c, Times\}_{K_{s,kdc}}$
   $Authenticator_c = \{ID_c, Realm_c, T_{c1}\}_{K_{c,tgs}}$

5. $C \rightarrow S: \quad Options, Ticket_s, Authenticator_c$
6. $S \rightarrow C: \quad \{T_{c2}, Subkey, \#Seq\}_{K_{c,s}}$
   $Authenticator_c = \{ID_c, Realm_c, T_{c2}, Subkey, \#Seq\}_{K_{c,s}}$

Como se mencionó en la sección anterior, los dos primeros mensajes corresponden al intercambio con el servidor de autenticación. En el mensaje (1), el cliente se identifica ante el *AS* solicitando un *Ticket* de otorgamiento de *ticket*, por lo que envía los siguientes elementos:

- *Options*: Es el campo de opciones. Tiene la finalidad de indicar mediante el uso de banderas (flags) algunas configuraciones para el *ticket* que el cliente recibirá.

- ID_c e ID_{tgs}: Es el identificador del cliente y del servidor con el cual el usuario desea comunicarse, en este caso el *TGS*.
- $Realm_c$: Dominio Kerberos al que pertenece el usuario.
- $Times$: Es el campo de tiempos. Esta compuesto por tres elementos, "*from*" que indica el tiempo de inicio de vigencia del *ticket* solicitado; "*till*" que es utilizado para señalar el tiempo de expiración del *ticket*; y "*rtime*", el cual es el tiempo para renovar el *ticket*.
- N_{c1}: Reto a ser respondido por el *AS*.

Una vez que *AS* recibe este mensaje, selecciona una llave de sesión $K_{c,tgs}$, y genera un *ticket* ($Ticket_{tgs}$) para el usuario. El *ticket* identifica al cliente, incluyendo datos como el ID_c, $Realm_c$, el campo de tiempos ($Times$) basado en los valores enviados por el cliente en el mensaje (1) además de la siguiente información nueva:

- $Flags$: Son las banderas que indican algunas características de funcionalidad del *ticket*. Más adelante se hablará un poco acerca de ellas.
- $K_{c,tgs}$: Llave de sesión para que el *TGS* pueda proteger la información que transmite con el cliente.
- AD_c: Dirección IP de la estación de trabajo del usuario.

La información contenida dentro del *ticket* se encuentra cifrada con la llave $K_{tgs,kdc}$, con la finalidad de proteger la llave de sesión. Esto indica que sólo el *TGS* podrá leer el *ticket*. Habiendo generado el *ticket*, *AS* ensambla el mensaje (2) añadiéndole los siguientes datos:

- ID_c y $Realm_c$: Identificador y dominio Kerberos del usuario.
- $Ticket_{tgs}$: Es el *ticket* recién generado por *AS*.
- Información protegida con la llave $K_{c,kdc}$: Contiene algunos campos con los que el usuario valida que el *ticket* fue generado en respuesta a su petición. Entre los datos que destacan son la llave de sesión $K_{c,tgs}$, la cual utilizará para comunicarse con el *TGS*; N_{c1}, que en este caso es la respuesta al reto enviado por el cliente en (1).

Después de que el cliente recibe el mensaje (2), contará con dos elementos importantes para poder dirigirse con el *TGS*, el $Ticket_{tgs}$ y la llave $K_{c,tgs}$. Estas credenciales deberán de ser guardadas y protegidas dentro de la estación de trabajo del usuario. Cuando el usuario requiera los servicios de algún servidor de la red, procederá a realizar el intercambio de mensajes con el Servidor de Otorgamiento de *Ticket* para obtener un *ticket* de servicio. El cliente prepara y envía el mensaje (3) con la siguiente información:

- *Options* y *Times*: Al igual que en la primera etapa, el cliente indica algunas configuraciones que desea se incluyan en el *ticket* que recibirá.
- ID_s: Identificador del servidor del cual el cliente desea obtener el servicio.
- N_{c2}: Un nuevo reto para ser contestado por el *TGS*.
- $Ticket_{tgs}$: El *ticket* obtenido en el intercambio anterior para que el cliente proporcione la llave $K_{c,tgs}$ al *TGS*.
- $Authenticator_c$: Es un conjunto de datos cifrados con la llave de sesión $K_{c,tgs}$ con los cuales el cliente se identifica ante el *TGS*. Contiene el ID_c y el $Realm_c$, así como una estampa de tiempo T_{c1} que el *TGS* valida de acuerdo a su conocimiento del tiempo.

El Identificador ($Authenticator_c$) tiene el propósito de ser utilizado una sola vez por lo que posee un tiempo de vida muy corto. Si por algún motivo el cliente no puede completar la etapa dos del protocolo, y desea volver a intentar establecer contacto con el *TGS*, deberá generar un nuevo identificador. El *TGS* recibe el mensaje (3), valida al usuario mediante los datos recibidos, y de igual manera que cuando el *AS* recibe el mensaje (2), genera una llave de sesión $K_{c,s}$ y un *ticket* ($Ticket_s$) para el usuario. El *TGS* forma el mensaje (4) siguiendo una estructura idéntica a la del mensaje (2) y se lo envía al cliente. En esta ocasión, el *ticket* estará cifrado con la llave $K_{s,kdc}$, de tal manera que sólo el *S* podrá leerlo. La llave de sesión $K_{c,s}$ dirigida para el cliente a su vez será protegida con la llave de sesión $K_{c,tgs}$.

Cuando el cliente recibe el mensaje (4), verifica la información contenida y procede a establecer el intercambio de autenticación con el servidor que le proporcionará el servicio. En el mensaje (5), el cliente le envía a *AS* los siguientes parámetros:

- *Options*: Dentro de estas opciones, el usuario puede solicitar que se lleve a cabo la autenticación mutua.
- $Ticket_s$: Contiene la llave de sesión $K_{c,s}$ para la comunicación segura entre el cliente y el servidor.
- $Authenticator_c$: Es un conjunto de datos que identifican al usuario (Identificador). Al igual que el Identificador presentado en la etapa de intercambio con el *TGS*, este contiene el ID_c, el $Realm_c$ y una estampa de tiempo T_{c2}. Adicionalmente, cuenta con dos parámetros opcionales, una sub-llave de sesión ($Subkey$) y un número de secuencia ($\#Seq$). La sub-llave de sesión se utiliza para proteger la información durante una sesión entre ambas entidades en lugar de la llave $K_{c,s}$. El número de secuencia es empleado para numerar los mensajes enviados por el servidor y evitar de mejor manera los ataques de repetición.

El servidor recibe la información del usuario, hace las validaciones necesarias y en caso de requerirse de autenticación mutua, responde con el mensaje (6). Este mensaje se encuentra constituido por la estampa de tiempo recibida en el Identificador (T_{c2}), la posible subllave de sesión confirmada (*Subkey*), y el número de secuencia con el que el cliente iniciará sus mensajes para esta sesión.

Cuando el proceso se ejecuta con éxito, el cliente y el servidor además de establecer una relación de confianza mutua, quedarán a disposición de una llave de cifrado simétrico para proteger la información que intercambien posteriormente. Con ello, se establecerá un canal de comunicación seguro.

Una funcionalidad adicional que provee la versión 5 de Kerberos con respecto a la versión 4 es la del uso de banderas (flags) dentro de los *tickets*. El uso de estas banderas permite en los *tickets* indicar algunas características de utilidad, expandiendo su funcionalidad. Algunas de las banderas se presentan en la tabla 3.1. En [31] se puede encontrar una explicación detallada de ellas.

PRE-AUTHENT	Indica que antes de que el AS emita un ticket, autenticó al usuario. El objetivo de tener pre-autenticación es el de evitar de que el KDC emita tickets sin verificar a los usuarios que los solicitan.
RENEWABLE	Le indica al TGS que este ticket puede ser utilizado para obtener un ticket de reemplazo que expirará más tarde.
PROXY	Indica que este ticket es un proxy.
FORWARDABLE	Le dice al TGS que un nuevo ticket de otorgamiento de ticket con una dirección de red diferente debe ser emitido en base a este ticket de otorgamiento de ticket.

Tabla 3.1: Banderas de la versión 5 de Kerberos

3.1.4. Pruebas físicas del protocolo Kerberos

Para la realización de las pruebas físicas de Kerberos, se procedió a montar un escenario como el que se muestra en la figura 3.2. El objetivo de esta prueba es demostrar el uso del protocolo Kerberos versión 5 para obtener el acceso a un servicio dentro de una red. En este caso se trata del acceso a un servicio de Eco (Echo en inglés) con autenticación previa.

Como se puede observar en la figura, el escenario cuenta con tres elementos, el usuario cliente, el servidor que proporciona el servicio de Eco, y el servidor Kerberos (también conocido como Centro de Distribución de Llaves o KDC). Para la aplicación cliente y la aplicación servidor se procedió a programar ambas mediante el uso del lenguaje de programación Python. Con respecto al servidor Kerberos, se utilizó la implementación desarrollada en el MIT para este servidor y se ejecutó bajo un ambiente Linux.

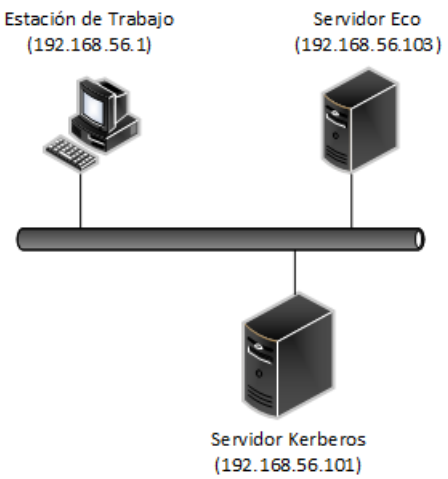

Figura 3.2: Escenario de pruebas de Kerberos

El primer paso de este experimento es instalar y configurar el servidor Kerberos en una máquina (computadora) con la finalidad de que pueda emitir *tickets*. La aplicación del servidor Kerberos puede ser encontrada para distintas versiones de sistemas operativos en la página web de Kerberos del MIT [38]. En cuestión de la configuración, es necesario indicar algunos parámetros como los son el nombre de nuestro dominio Kerberos así como crear la base de datos en donde se registraran a los usuarios del sistema. Para este experimento, se creó un dominio Kerberos llamado "CINVESTAV.MX" y dos clientes del sistema Kerberos, "redes3" que es el usuario que desea acceder a un servicio (o servicios) dentro de la red, y "echo/server.cinvestav.mx" que el servidor que proporciona el servicio Eco. Una mayor referencia acerca de las configuraciones necesarias del servidor Kerberos se puede encontrar en [39].

Una vez configurado el servidor Kerberos, se procede a configurar las estaciones de trabajo (o servidores) de los usuarios del sistema Kerberos. Dentro de estas configuraciones se encuentra el hacer que la estación de trabajo proceda de forma automática a comunicarse con el Servidor de Autenticación (contenido dentro del servidor Kerberos) una vez que el usuario ha ingresado su identificador y su contraseña para iniciar sesión. Con esto, un usuario podrá disponer de un "*Ticket* de otorgamiento de *ticket*" que será utilizado por las distintas aplicaciones cliente para acceder a los servicios de la red sin la necesidad de que introduzca nuevamente sus credenciales. A esta característica se le conoce en inglés como "*Single Sign On*". Para el caso de las máquinas que proporcionan los servicios (usuarios servidores), se configuran de forma tal que el proceso de introducir un identificador y una contraseña se realice de forma automática, ya que estas estaciones operan por lo general sin la ayuda de alguna interacción humana. En [39] se puede encontrar la referencia de como realizar estas

configuraciones.

Teniendo el sistema configurado, para que una aplicación cliente y una aplicación servidor puedan utilizar la autenticación Kerberos, es necesario agregarles dentro de su código cierta cantidad de líneas que realicen el proceso de autenticación por este método. A las aplicaciones que incluyen código para ejecutar la autenticación Kerberos se les da el nombre de "Aplicaciones Kerberizadas". Tanto a la aplicación de Eco cliente como a la aplicación de Eco servidor implementadas fueron Kerberizadas para poder llevar a cabo el proceso de Kerberos. Esto se realizó con la ayuda de una Interfaz de Programación de Aplicaciones conocida como GSSAPI (*Generic Security Service Application Program Interface*) la cual se encuentra especificada en el RFC 2743 [40]. El diagrama de flujo para las aplicaciones cliente y servidor se muestra en la figura 3.3.

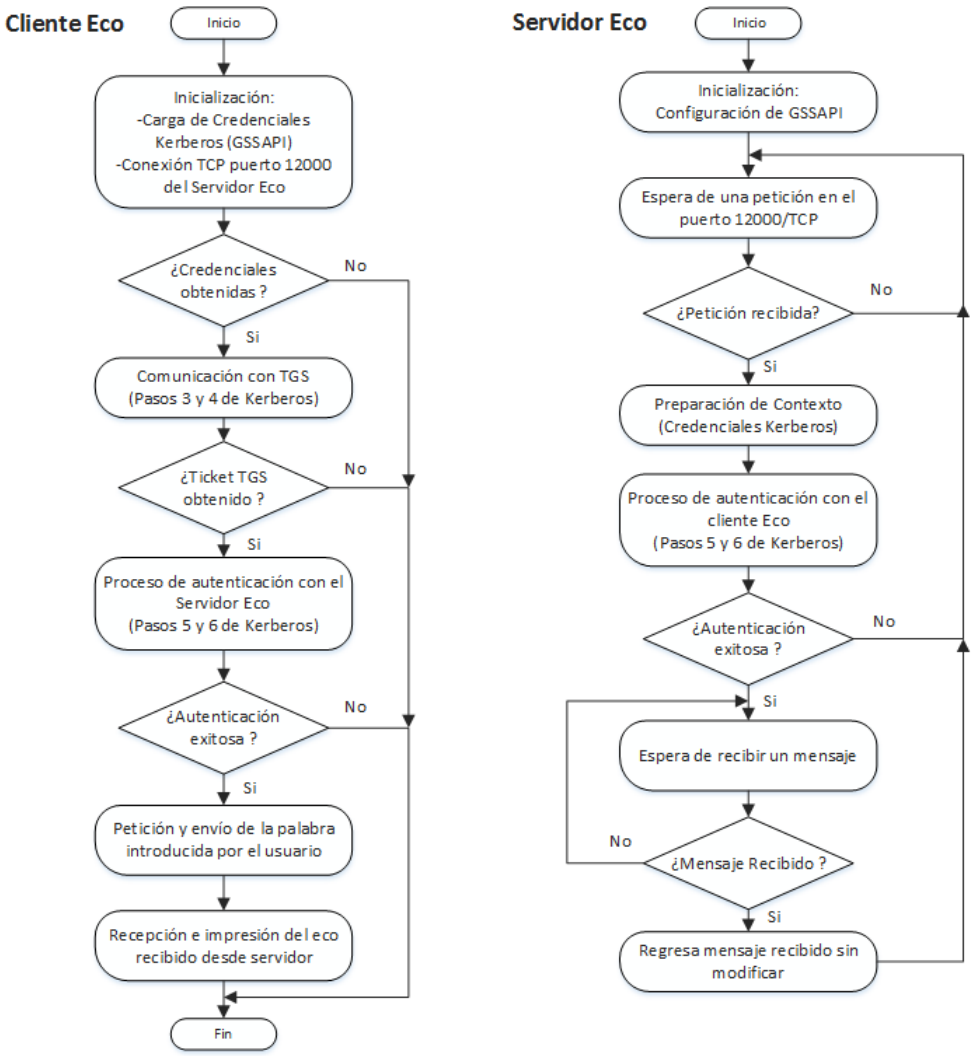

Figura 3.3: Diagramas de flujo de las aplicaciones cliente y servidor de eco

Finalmente, se procede a ejecutar las aplicaciones de Eco. En primer lugar se ejecuta la aplicación servidor de Eco para que escuche peticiones por un puerto designado para uso general (en este caso se utilizó el puerto 12000/tcp). Luego de esto, se ejecuta la aplicación cliente de Eco, la cual procede en primera instancia a obtener un "*Ticket* de otorgamiento de servicio" para el servicio de Eco comunicándose con el Servidor de Otorgamiento de *Ticket* (ubicado dentro del servidor Kerberos). Teniendo este *ticket*, el programa lo envía al servidor de Eco, el cual lo valida y concede el permiso al usuario para el servicio de Eco. A partir de este momento, la aplicación cliente le solicita al usuario escribir una palabra la cual será enviada y regresada son modificar por el servidor de Eco.

En la figura 3.4, se muestran el tráfico de datos obtenidos con la herramienta Wireshark para una ejecución del proceso de Kerberos (junto con la aplicación de Eco). Las líneas número tres y cuatro corresponden al intercambio que realiza el cliente con el Servidor de Autenticación (*AS*). En primera instancia, el cliente (192.168.56.1) envía una petición al *AS* (192.168.56.101). El AS responde con el mensaje AS-REP, en el cual incluye todos los parámetros del mensaje dos del protocolo tal como se especificó en la descripción de Kerberos (ver sección 3.1.3). Es importante recordar que este intercambio ocurre cuando el usuario accede a su estación de trabajo e ingresa su ID de usuario y contraseña, es decir, antes de que se ejecute la aplicación de Eco. Las líneas diez y once muestran el intercambio entre la aplicación cliente del usuario con el Servidor de Otorgamiento de *Ticket* (*TGS*). Aquí, la aplicación cliente corresponde a la aplicación de Eco Kerberizada. También en la captura realizada con Wireshark, se puede apreciar que el cursor apunta a la línea once, mostrándose en la parte de abajo de la figura los datos correspondientes al mensaje de respuesta del servidor *TGS* (TGS-REP). Se puede ver que los datos principales presentes en este mensaje corresponden de forma fiel al mensaje cuatro del protocolo Kerberos mostrado en el apartado anterior.

Las líneas doce y catorce de la captura corresponden al mensaje cinco y seis del protocolo Kerberos respectivamente. Debido al uso de GSSAPI, estos dos mensajes se encuentran encapsulados dentro de los mensajes generados por GSSAPI, por lo que no es posible ver su formato de forma directa con el programa Wireshark. En esta etapa, la comunicación corresponde a la establecida por la entidad cliente Eco (192.168.56.1) y la entidad servidor Eco (192.168.56.103). Las líneas dieciséis y diecisiete constituyen el intercambio efectuado para la aplicación de Eco. El cliente envía como mensaje la palabra "Hola", y el servidor responde regresando la palabra recibida sin modificación alguna.

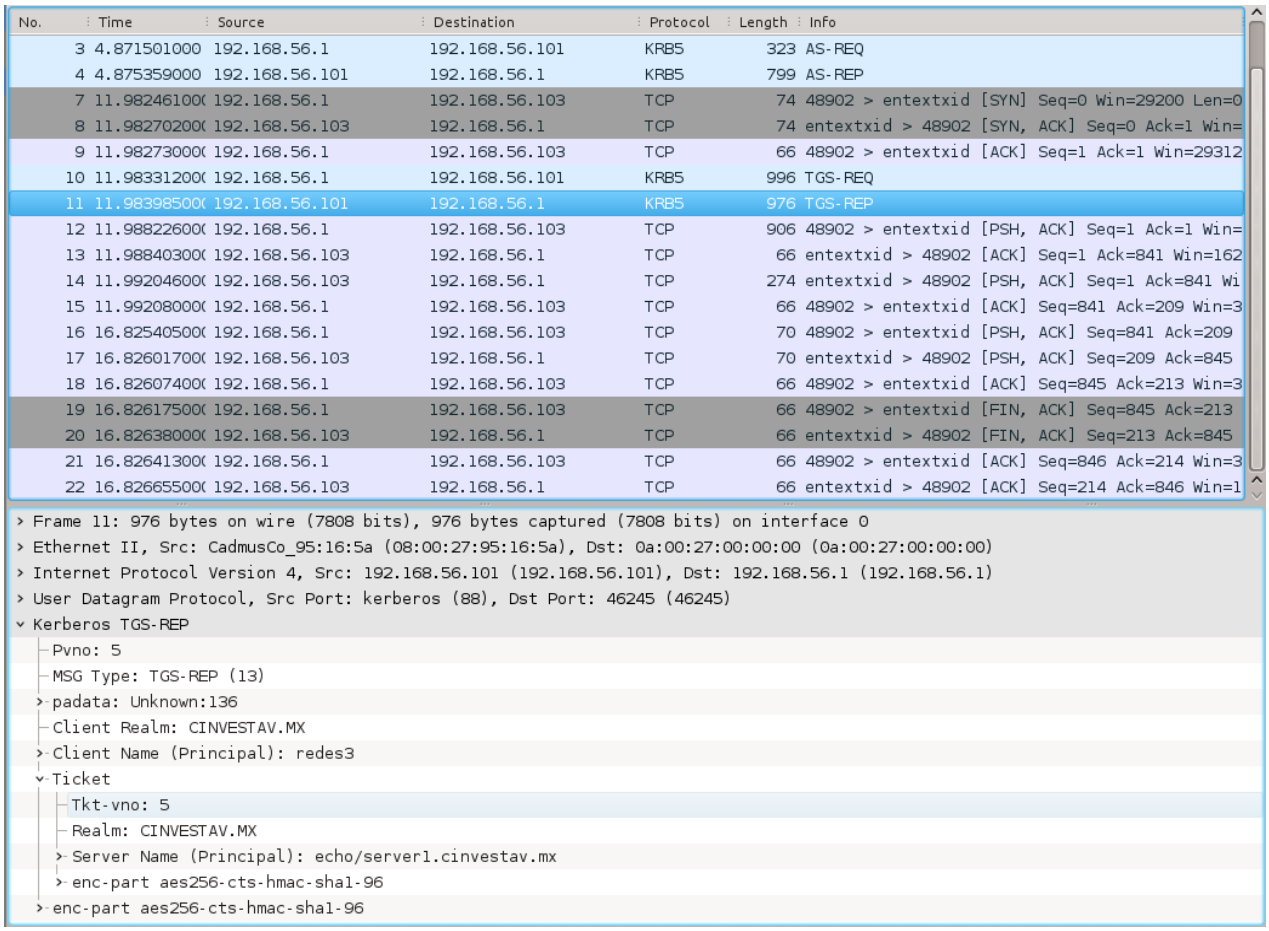

Figura 3.4: Captura de la ejecución de un proceso de autenticación en Kerberos

3.2. RADIUS

En esta sección se estudia y analiza el protocolo RADIUS. Primero, se lleva a cabo una introducción al mismo describiendo los conceptos principales y planteando el escenario en el que opera. Luego de esto, se describe el framework conocido como Protocolo de Autenticación Extensible (EAP por sus siglas inglés), el cual se encuentra íntimamente relacionado al protocolo RADIUS para proporcionar diversos mecanismos de autenticación. En tercer lugar, se proporciona la descripción del funcionamiento del protocolo RADIUS, en base a la especificación expuesta en su respectivo RFC. Finalmente, se presenta la prueba de la implementación realizada para este protocolo.

3.2.1. Introducción al protocolo RADIUS

RADIUS (*Remote Authentication Dial-In User Service* en inglés) es un protocolo perteneciente a una familia de protocolos conocida como AAA (*Authentication, Authorization, and Accounting* en inglés) los cuales permiten gestionar de manera centralizada servicios de "autenticación, autorización y registro" para usuarios que desean acceder a los recursos de una red. Fue creado por Livingston Enterprises en 1991 en respuesta a la necesidad que se presentaba en la compañía Merit Network (compañía proveedora de internet sin fines de lucro) de administrar de una forma creativa el acceso por marcación (*dial-in* en inglés) a varios de los puntos de presencia localizados a través de su red. Posteriormente se convirtió en un estándar IETF [41]. Sus especificaciones se encuentran en el RFC 2865 y en el RFC 2866, además existen una gran variedad de extensiones especificadas en diversos RFC's. Actualmente cuenta con un gran uso y aceptación dentro de los sistemas de cómputo distribuido debido a su simplicidad.

Como se explicó en el capítulo uno, el término autenticación hace referencia al proceso por el cual se determina si un usuario tiene permiso para acceder a un servicio de la red mediante la presentación y validación de credenciales. Los dos términos restantes de AAA, se explican a continuación:

- **Autorización:** Es el proceso de verificar que un usuario ya autenticado tiene la autoridad para efectuar una determinada operación (otorgar privilegios). Ejemplo de esto es, seleccionar la calidad del servicio que se le brindará al usuario de acuerdo a sus privilegios.

- **Registro:** Se refiere al proceso de llevar un registro del consumo de recursos que realizan los usuarios. El registro suele incluir aspectos como la identidad del usuario, la naturaleza del servicio prestado, y cuando empezó y terminó el uso de dicho servicio.

RADIUS se basa en un modelo de tres participantes (o entidades), los cuales son [12]:

- **Peticionario (Suplicant en inglés).** Es el usuario que intenta obtener acceso a la red. Puede ser por ejemplo un individuo que intenta conectarse a una red inalámbrica mediante una computadora utilizando el estándar IEEE 802.11 (Wi-Fi).
- **Servidor de Acceso de Red (NAS por sus siglas en inglés).** Es la entidad que interactúa directamente con el usuario peticionario. Para el caso de la arquitectura de control de acceso IEEE 802.1X, esta entidad puede estar representada por un conmutador o por un punto de acceso inalámbrico. En RADIUS, esta entidad es un cliente, por lo que se le denomina cliente RADIUS.
- **Servidor de autenticación (SA).** Es la entidad que posee la autoridad y la información necesaria para decidir si se le otorga acceso a un usuario. En RADIUS, este elemento es conocido como servidor RADIUS.

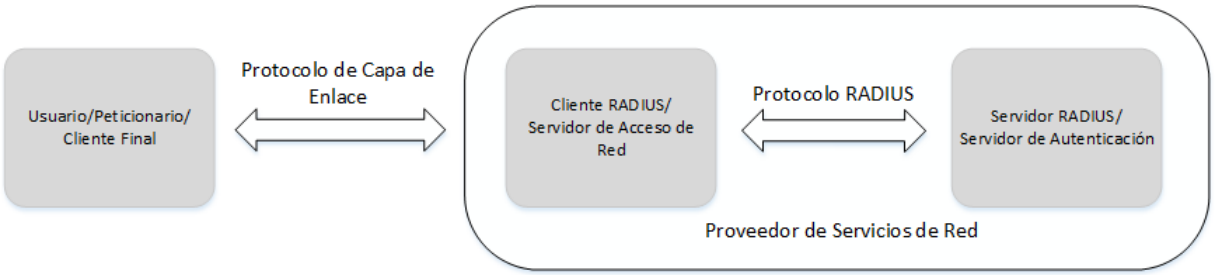

Figura 3.5: Modelo de autenticación RADIUS

En la figura 3.5 se puede ver la interacción entre estos tres elementos. Básicamente esta interacción es como sigue: el usuario peticionario establece comunicación con el *NAS* para enviar sus credenciales, y el *NAS* replica esta información hacia el *SA*. La comunicación entre el usuario y el *NAS* es a nivel capa de enlace (mediante Ethernet o Wi-Fi por ejemplo), mientras que la comunicación entre el *NAS* y el *SA* se lleva a nivel capa de aplicación (mediante el protocolo RADIUS). Esta arquitectura cuenta con la ventaja de que se pueden disponer de varios puntos de acceso a una red sin la necesidad de que cada uno de ellos cuente con un registro de los usuarios que tienen derecho a usar el sistema.

El servidor RADIUS puede soportar una variedad de métodos para autenticar usuarios. Entre estos principalmente se tienen los métodos de presentar un identificador y una contraseña, como lo son PAP (*Password Authentication Protocol* en inglés) y CHAP (*Challenge Handshake Authentication Protocol* en inglés). También permite el uso de métodos más sofisticados, como lo es el Protocolo de Autenticación Extensible (EAP por sus siglas en inglés).

EAP es el método más utilizado en la actualidad debido a que permite el uso de una gama amplia de protocolos de autenticación.

3.2.2. El Protocolo de Autenticación Extensible (EAP)

El Protocolo de Autenticación Extensible es un framework de autenticación que encapsula métodos múltiples de autenticación. Se encuentra definido en el RFC 3748 y puede ser empleado en varias tecnologías de nivel de capa de enlace como son PPP (*Point to Point Protocol* en inglés), Ethernet, WLAN y otros. En la figura 3.6 se ilustra las capas de protocolos que forman parte del contexto de EAP [42].

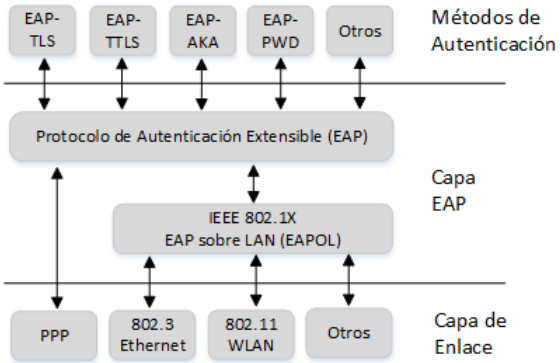

Figura 3.6: Capas del contexto EAP

Se puede apreciar que en la capa superior se encuentran los métodos múltiples de los que EAP puede hacer uso. En la actualidad existen una gran variedad de métodos de autenticación para EAP muchos de los cuales se encuentran estandarizados en diversos documentos RFC. Entre los métodos más comunes (y de mayor uso en la actualidad) tenemos a EAP-TLS (RFC 5216), EAP-TTLS (RFC 5281), EAP-IKEv2 (RFC 5106), EAP-AKA (RFC 4187) y EAP-PWD (RFC 5931).

Existen dos tipos de modelos de operación para EAP especificados en el RFC 3748, estos son: el modelo "*pass-through*" (pasa a través de) y el modelo de "*multiplexing*" (multiplexaje) [43]. En el modelo "*pass-through*" existen tres entidades involucradas en la autenticación EAP, el Peticionario, el Servidor de Acceso de Red (*NAS*) y el Servidor de Autenticación (*SA*). El Peticionario se establece en el dispositivo que utiliza el usuario, el *NAS* puede ser un dispositivo como un conmutador o punto de acceso inalámbrico (que actúan como repetidores) y el *SA* por lo general reside en servidores AAA, como lo es RADIUS. Este modelo es el más ampliamente utilizado y en el cual nos enfocaremos.

En el modelo de "multiplexaje", existen sólo dos entidades independientes. Tanto el *NAS* como el *SA* se encuentran ubicados en el mismo dispositivo, mientras que el Peticionario

continúa siendo una entidad aparte. Este modelo es más aplicable para redes ad-hoc.

El formato de un mensaje EAP incluye los campos mostrados en la figura 3.7 [44].

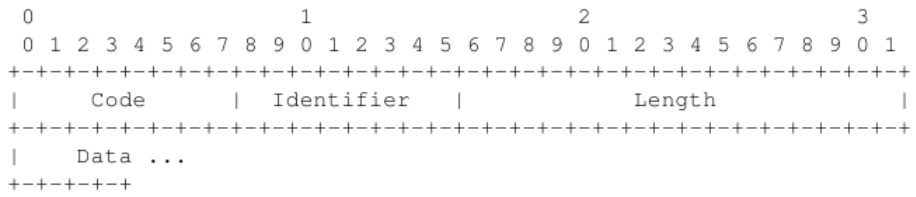

Figura 3.7: Estructura de un mensaje en EAP.

- *Code* (Código): Octeto que identifica el tipo de mensaje EAP. Los códigos son *Request* (1), *Response* (2), *Success* (3), y *Failure* (4).
- *Identifier* (Identificador): Octeto utilizado para enlazar Peticiones (*Requests*) con Respuestas (*Responses*).
- *Length* (Longitud): Son dos octetos que indican la longitud, en octetos, del mensaje EAP, incluyendo los campos de Código (*Code*), Identificador (*Identifier*), Longitud (*Length*), y Datos (*Data*).
- *Data* (Datos): Conjunto de octetos que contienen la información para llevar a cabo la autenticación. Típicamente este campo consiste de un subcampo de Tipo (*Type*), indicando el tipo de datos contenido (el protocolo de autenticación a utilizar), y otro campo conocido como *Type-Data* en donde se colocan los datos para un protocolo en específico.

Los mensajes de Éxito (*Success*) y Fallo (*Failure*) no incluyen el campo de datos. El intercambio de mensajes en EAP procede como sigue. Después de que el intercambio de mensajes de bajo nivel establece la necesidad de un intercambio EAP, el *NAS* envía una Petición de identidad (*Request/Identity*) al usuario que desea autenticarse, por lo que este usuario envía una Respuesta (*Response*) que contiene información de identidad (como nota, a partir de este momento en el caso del modelo "*pass-through*", el *NAS* opera sólo como un repetidor por lo que envía toda la información del cliente hacia el *SA*). Esto es seguido por una secuencia de Peticiones-Respuestas por parte de las entidades involucradas intercambiando información de autenticación. La información intercambiada y el número de Peticiones-Respuesta dependen del método de autenticación. La conversación continúa hasta que ocurra uno de los siguientes casos: el *SA* determine que no puede autenticar al usuario y por lo tanto transmite el mensaje *EAP Failure*; o el *SA* determine que ha ocurrido una autenticación exitosa por lo que emite el mensaje *EAP Success*.

3.2.3. Descripción del protocolo RADIUS

Como se mencionó, el protocolo RADIUS se encuentra especificado principalmente en dos documentos RFC, el primero es el RFC 2865 [45] en donde se explica la parte de autenticación y autorización del protocolo, mientras que el segundo es el RFC 2866 [46] y en este se detalla la parte de registro (*accounting*). La descripción presentada a continuación esta basada en el RFC 2865 por lo que no se tomará en cuenta la parte de registro en RADIUS ya que se encuentra fuera del objetivo de este trabajo.

Cuando un cliente (NAS) es configurado para utilizar RADIUS, cualquier usuario del cliente (Peticionario) presenta información de autenticación a esta entidad. Una vez que el cliente ha obtenido dicha información, este procede a autenticarse empleando el protocolo RADIUS. Para ello, el cliente crea un mensaje de petición de acceso (*Access-Request* en inglés) conteniendo datos (atributos) como el nombre del usuario y su contraseña, el ID del cliente, el ID del puerto por el cual el usuario está accediendo, entre otros.

La petición de acceso se envía al servidor RADIUS a través de la red. Una vez que la recibe, el servidor RADIUS en primera instancia valida al cliente que envió la información (mediante el uso de un secreto compartido entre ambos participantes). Si el cliente es válido, el servidor RADIUS consulta una base de datos que contiene los requerimientos que se deben cumplir para que se le permita el acceso al usuario. Entre estos requerimientos se encuentra la verificación de la contraseña recibida, pero pueden existir algunos otros.

Si alguna de las condiciones no se cumplen, el servidor RADIUS envía una respuesta de acceso denegado (*Access-Reject* en inglés) la cual indica que la petición del usuario es inválida. En esta respuesta el servidor RADIUS puede enviar un mensaje indicando el porqué no se le permitió el acceso al usuario.

Si todas las condiciones se cumplen, pero el servidor RADIUS aun requiere de mas información por parte del usuario, el servidor RADIUS envía una respuesta de reto (*Access-Challenge* en inglés). Esta respuesta puede incluir información de estado entre otros datos. Una vez que el cliente recibe este mensaje, procede a solicitar al usuario la información requerida por el servidor RADIUS y genera un nuevo mensaje de "*Access-Request*" en el cual añade los datos recientes obtenidos del usuario en cuestión. De igual manera, el servidor RADIUS recibe el nuevo "*Access-Request*" y procede a hacer las validaciones necesarias de tal forma que si alguna condición necesaria no se cumple, enviará un mensaje de "*Access-Reject*". En caso de aún requerir más información, el servidor RADIUS volverá a enviar un mensaje de "*Access-Challenge*".

Si todas las condiciones se cumplen y no se requiere información adicional, el servidor RADIUS envía una lista de parámetros de configuración en el mensaje de acceso permitido

(*Access-Accept*). Estos parámetros pueden incluir el tipo de servicio a proporcionar y todos los valores necesarios para poder otorgarle dicho servicio al usuario.

El protocolo RADIUS emplea el protocolo de capa de transporte UDP y tiene asignado el puerto 1812 para autenticación/autorización. El formato de un mensaje RADIUS contiene los campos mostrados en la figura 3.8.

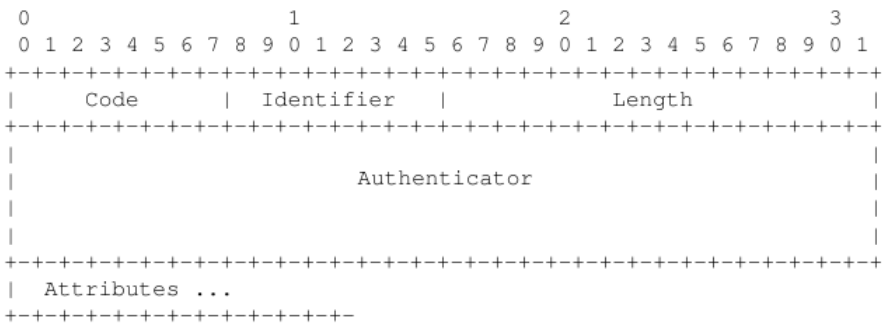

Figura 3.8: Estructura de un mensaje en el protocolo RADIUS

- *Code* (Código): Es un octeto que identifica el tipo de paquete RADIUS. Los valores principales para el proceso de autenticación son *Access-Request* (1), *Access-Accept* (2), *Access-Reject* (3) y *Access-Challenge* (11). Existen otros valores que se ocupan por ejemplo para el proceso de registro.

- *Identifier* (Identificador): Es un octeto que tiene la finalidad de asociar peticiones con respuestas.

- *Length* (Longitud): Dos octetos que indican la longitud del mensaje. Incluye los campos de *Code*, *Identifier*, *Length*, *Authenticator* y *Attributes*.

- *Authenticator*: Se encuentra constituido por dieciséis octetos y se utiliza para autenticar la respuesta del servidor.

- *Attributes* (Atributos): Es el conjunto de octetos que constituye la información a enviar en cada uno de los mensajes del protocolo.

Los atributos se encuentran especificados mediante el formato presentado en la figura 3.9.

- *Type* (Tipo): Es un octeto que define el tipo de información contenida. Los valores numéricos que identifican a cada atributo se encuentran definidos en los diversos documentos RFC's que tratan acerca del protocolo RADIUS.

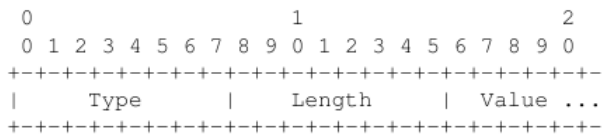

Figura 3.9: Formato para los atributos dentro del protocolo RADIUS

- *Length* (Longitud): Es un octeto que indica la longitud total del atributo (incluyendo este campo así como el de *Type* y *Value*).

- *Value* (Valor): Es un conjunto de octetos que contienen la información propia del atributo. De acuerdo al tipo de atributo, los octetos que conforman este campo presentan distintos formatos como pueden ser cadenas de caracteres, números enteros, valores binarios, entre otros.

Existe un cantidad diversa de atributos en la actualidad. Aunque el campo "*Type*" sólo cuenta con un tamaño de un byte para definir tipos diferentes de atributos (lo cual implica que sólo podrían existir 256 atributos distintos), se han hecho algunas extensiones al protocolo para poder agregar más atributos a RADIUS. Algunos de los atributos más importantes (y que se utilizarán en las pruebas físicas de este protocolo) se muestran en la tabla 3.2:

User-Name	Indica el nombre del usuario a ser autenticado.
NAS-IP-Address	Contiene la dirección IP del servidor de acceso a la red.
NAS-Port	Contiene el número de puerto físico del NAS que está autenticando al usuario.
Calling-Station-Id	Indica la dirección MAC del dispositivo del usuario.
EAP-Message	Encapsula un mensaje del protocolo EAP.
Message-Authenticator	Contiene un código de integridad de mensaje de diesiseis octetos para proteger la integridad de los mensajes que contienen atributos del tipo "EAP-Message".

Tabla 3.2: Atributos comunes del protocolo RADIUS

3.2.4. Pruebas físicas de RADIUS

Para la realización de las pruebas físicas del protocolo RADIUS, se procedió a montar un escenario como el que se muestra en la figura 3.10. Con este escenario, se busca otorgar el acceso a la red a un dispositivo (como lo es un teléfono móvil) el cual hace uso de la tecnología Wi-Fi para comunicarse con un punto de acceso inalámbrico.

Como se puede observar de la figura, el dispositivo que desea el acceso a la red se autentica mediante el uso de EAP. Bajo este escenario, el punto de acceso funciona en modo "pass-through" por lo que encapsula los paquetes en formato EAP a el formato del protocolo

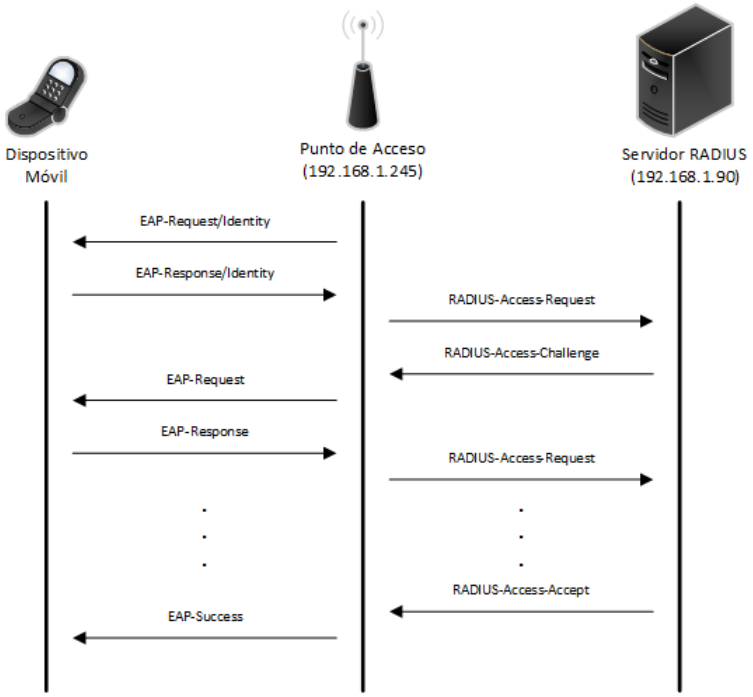

Figura 3.10: Escenario de pruebas e intercambio de mensajes del experimento

RADIUS para enviárselos al servidor de autenticación. Los pasos generales del proceso de autenticación son los siguientes:

1. El intercambio EAP inicia con el punto de acceso emitiendo un mensaje *EAP-Request/Identity* al usuario suplicante (que este caso se trata de un dispositivo móvil).

2. El móvil responde con un mensaje *EAP-Response/Identity*, el cual el punto de acceso recibe. El paquete es entonces encapsulado en el formato RADIUS y enviado al servidor RADIUS como un mensaje *RADIUS-Access-Request*.

3. El servidor AAA responde con un paquete *RADIUS-Access-Challenge*, el cual es pasado al móvil como un mensaje *EAP-Request*. Esta petición es del tipo apropiado del método de autenticación elegido y contiene información relevante que sirve como reto.

4. El dispositivo móvil genera un mensaje *EAP-Response* y lo envía al punto de acceso. La respuesta se traslada por el punto de acceso a un mensaje *RADIUS-Access-Request* que contiene la respuesta al reto realizado en la petición reciente. Los pasos 4 y 5 se pueden repetir múltiples veces de acuerdo al método EAP empleado.

5. El servidor AAA otorga el acceso con un mensaje *RADIUS-Access-Accept*. Al recibir este mensaje, el punto de acceso emite un paquete *EAP-Success*, el cual le indica al usuario que el proceso fue exitoso.

El servidor RADIUS utilizado para este experimento fue desarrollado bajo el lenguaje de programación Python siguiendo las especificaciones expuestas en [45]. A su vez, debido a que se eligió la autenticación por medio de EAP, fue necesario programar el servidor EAP embebido dentro del servidor RADIUS. Para el encapsulado de los mensajes EAP dentro del protocolo RADIUS, se hizo uso de los atributos "*EAP-Message*" y "*Message-Authenticator*" tal como se describe en el RFC 3579 [47]. El método EAP elegido para llevar a cabo estas pruebas es el EAP-PWD (*EAP-Password*) el cual requiere de seis mensajes para completar una autenticación entre dos entidades.

EAP-PWD se encuentra especificado en el RFC 5931 y tiene el objetivo de autenticar y generar llaves de sesión a partir del uso de una contraseña. Este protocolo permite el uso de contraseñas débiles (de baja entropía) para generar llaves criptográficas fuertes. Cuenta con la característica principal de que si la contraseña es comprometida, esta no provee de información alguna acerca de los secretos (llaves) generados con anterioridad en ejecuciones previas del protocolo, por lo que la información enviada previamente no se verá comprometida. A esta característica se le conoce en inglés bajo el término de "*forward secrecy*".

EAP-PWD utiliza la criptografía de logaritmo discreto para lograr la autenticación y el establecimiento de llaves. Permite el uso de dos técnicas criptográficas distintas para el enmascaramiento de la contraseña, la primera es la Criptografía de Campos Finitos (FFC por sus siglas en inglés) y la segunda es la Criptografía de Curvas Elípticas (ECC por sus siglas en inglés). Además, EAP-PWD emplea funciones tanto pseudoaleatorias como aleatorias para generar parámetros así como una función de generación de llaves. Dichas funciones se encuentran definidas en [48]. El intercambio de mensajes de EAP-PWD se muestra en la figura 3.11. En esta figura se omite el Servidor de Acceso de Red (NAS).

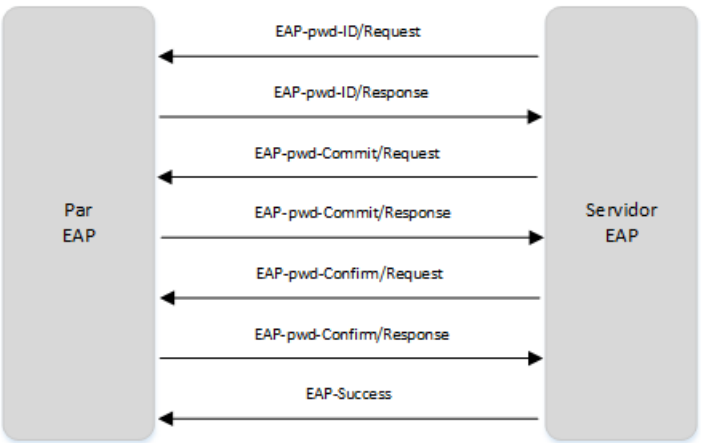

Figura 3.11: Intercambio de mensajes en EAP-PWD

El Par EAP (o peticionario EAP) y el Servidor utilizan el intercambio de identidad

(*EAP-pwd-ID/Request*, *EAP-pwd-ID/Response*) para descubrir la identidad de cada uno así como intercambiar y acordar los parámetros de la técnica criptográfica a emplear (FFC o ECC) en los mensajes consecuentes. En el intercambio de mensajes "*Commit*" (*EAP-pwd-Commit/Request*, *EAP-pwd-Commit/Response*), el Par EAP y el Servidor intercambian información para generar un secreto compartido y asocian la contraseña a un valor particular (valor aleatorio) de esta ejecución del protocolo. Finalmente, en el intercambio de confirmación (*EAP-pwd-Confirm/Request*, *EAP-pwd-Confirm/Response*), el Par EAP y el Servidor prueban que se encuentran activos y que poseen conocimiento de la contraseña mediante generación y validación de datos de verificación. Posterior al intercambio del método PWD se envía un mensaje de *EAP-Success* como se encuentra especificado en el protocolo EAP para la terminación.

El formato del paquete EAP junto con PWD es el mostrado en la figura 3.12.

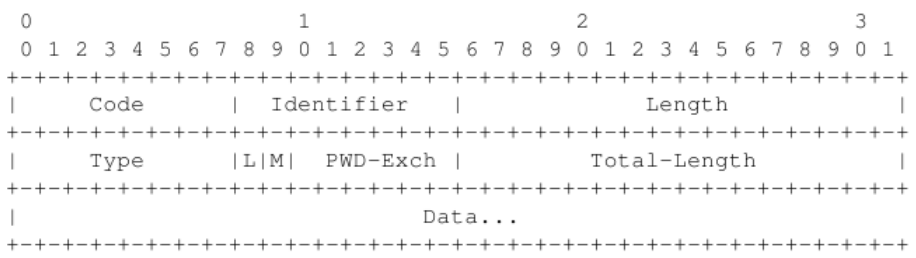

Figura 3.12: Estructura de un mensaje en EAP-PWD

Los primeros tres campos constituyen el encabezado EAP (los cuales fueron descritos en la sección 3.2.2). El campo "*Type*" es un octeto que sirve para indicar el método EAP a utilizar, que para el caso de PWD se utiliza el valor 52. El campo "*PWD-Exch*"es un octeto del cual se ocupan los cinco bits menos significativos para indicar si el mensaje es "*EAP-pwd-ID*", *"EAP-pwd-Commit"* o *"EAP-pwd-Confirm*". El campo "*Total-Length*" son dos octetos que se pueden usar para indicar la longitud total de los datos en caso de que se requiera de partición de paquetes (si no hay partición de paquetes este campo no se utiliza). En el campo "*Data*" se colocan los datos a enviar durante las tres etapas del protocolo. En cada etapa se utiliza un formato distinto para este campo. Una descripción más amplia del formato específico en cada etapa se puede encontrar en [48].

En este experimento se implementó la opción de criptografía de curvas elípticas (ECC) utilizando un campo finito (GF(p)) seguro basado en un número primo "p" de 256 bits. El valor del número "p" empleado se encuentra definido en [49] bajo el nombre de grupo diecinueve. Para analizar los paquetes enviados por el NAS (punto de acceso inalámbrico) hacia el servidor RADIUS, se utilizó la herramienta de análisis de paquetes Wireshark y los resultados obtenidos se muestran en la figura 3.13.

De la captura (figura 3.13), se puede observar con claridad el flujo de mensajes del protocolo RADIUS. El NAS (192.168.1.245) comienza la comunicación enviando una petición de acceso (*"Access-Request"*) al servidor RADIUS (192.168.1.90). El servidor RADIUS responde solicitando más información mediante un mensaje "*Access-Challenge*". Dentro del "*Access-Challenge*" se incluye la información en formato EAP correspondiente al mensaje "*EAP-pwd-ID/Request*". El intercambio de mensajes *"Access-Request/Access-Challenge"* continúa hasta que se cumplen los seis pasos especificados para el protocolo EAP-PWD, y termina con un mensaje adicional en donde se indica que el proceso terminó con éxito ("Access-*Accept*"). El mensaje de "Access-*Accept*" encapsula al mensaje "*EAP-Success*" en uno de los atributos del protocolo RADIUS.

En la captura, se muestra el cursor apuntando al segundo mensaje "*Access-Request*", observándose en la parte de abajo los atributos enviados por el NAS al servidor RADIUS. La cantidad de atributos enviados por el NAS depende del fabricante del punto de acceso, pero es común ver en la respuesta del NAS atributos tales como los que se presentan en la tabla 3.2. Los atributos "*EAP-Message*" y "*Message-Authenticator*" sólo aparecen cuando se ejecuta un proceso de autenticación mediante uno de los métodos EAP. Para este ejemplo se puede ver (de la captura) que se incluyen estos atributos y que se especifica que el método EAP que se utiliza es el EAP-PWD.

Con respecto a la comunicación entre el dispositivo móvil y el punto de acceso, podemos decir que cuenta con el mismo número de intercambio de mensajes que los que se muestran en la figura anterior. Esta se puede obtener colocando una computadora PC (con tarjeta inalámbrica incluida) que ejecute Wireshark en modo monitor en un lugar cercano que se encuentre dentro del área de cobertura del punto de acceso inalámbrico.

Finalmente, en las figuras 3.14, 3.15 y 3.16 se muestran los diagramas de flujo correspondientes a la aplicación realizada para el servidor RADIUS junto con el servidor EAP y el módulo EAP-PWD.

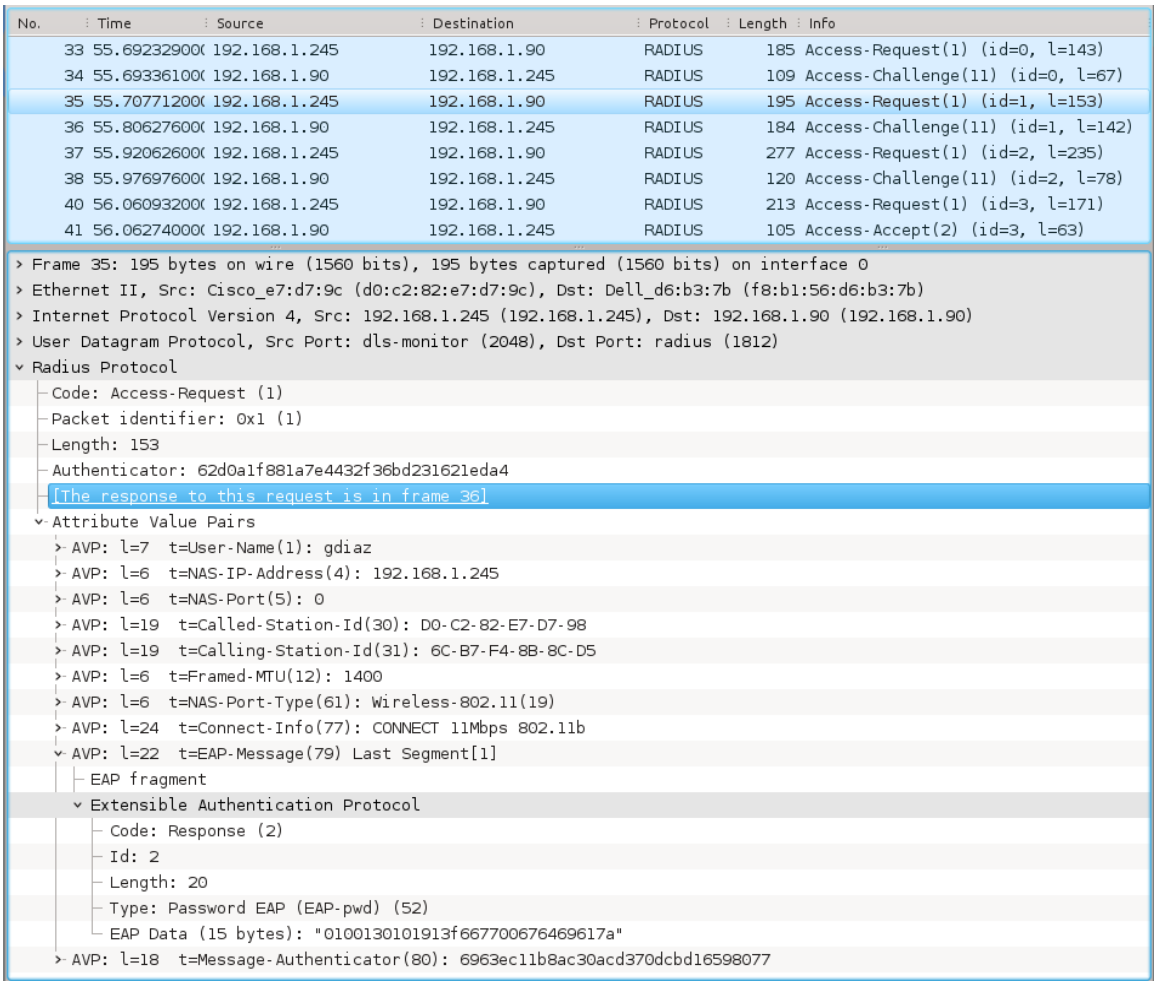

Figura 3.13: Captura de la ejecución de un proceso de autenticación en RADIUS

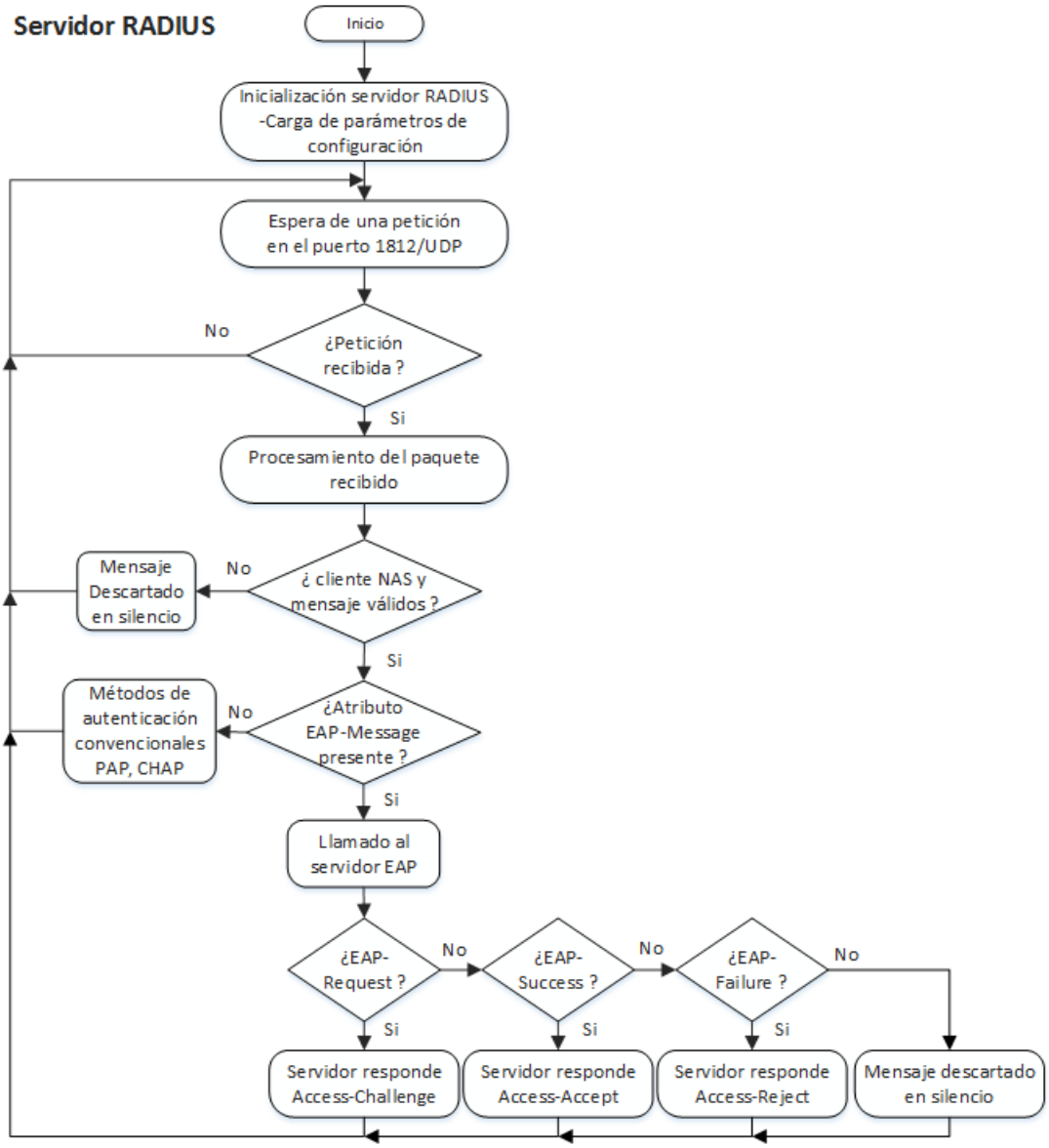

Figura 3.14: Diagrama de flujo del servidor RADIUS implementado

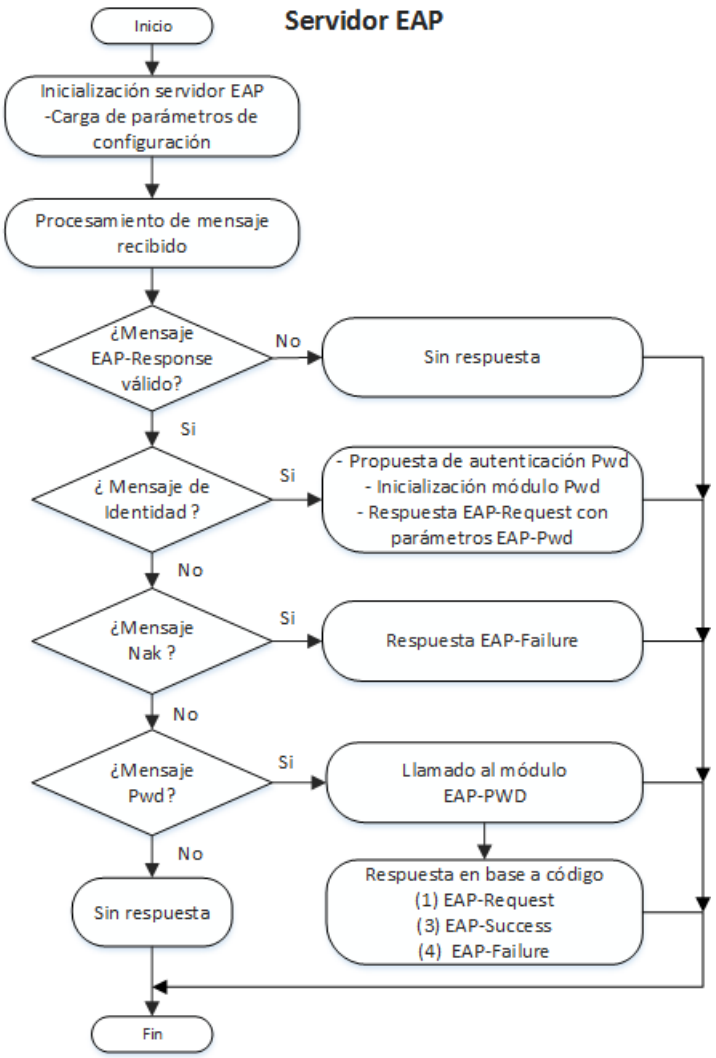

Figura 3.15: Diagrama de flujo del servidor EAP implementado

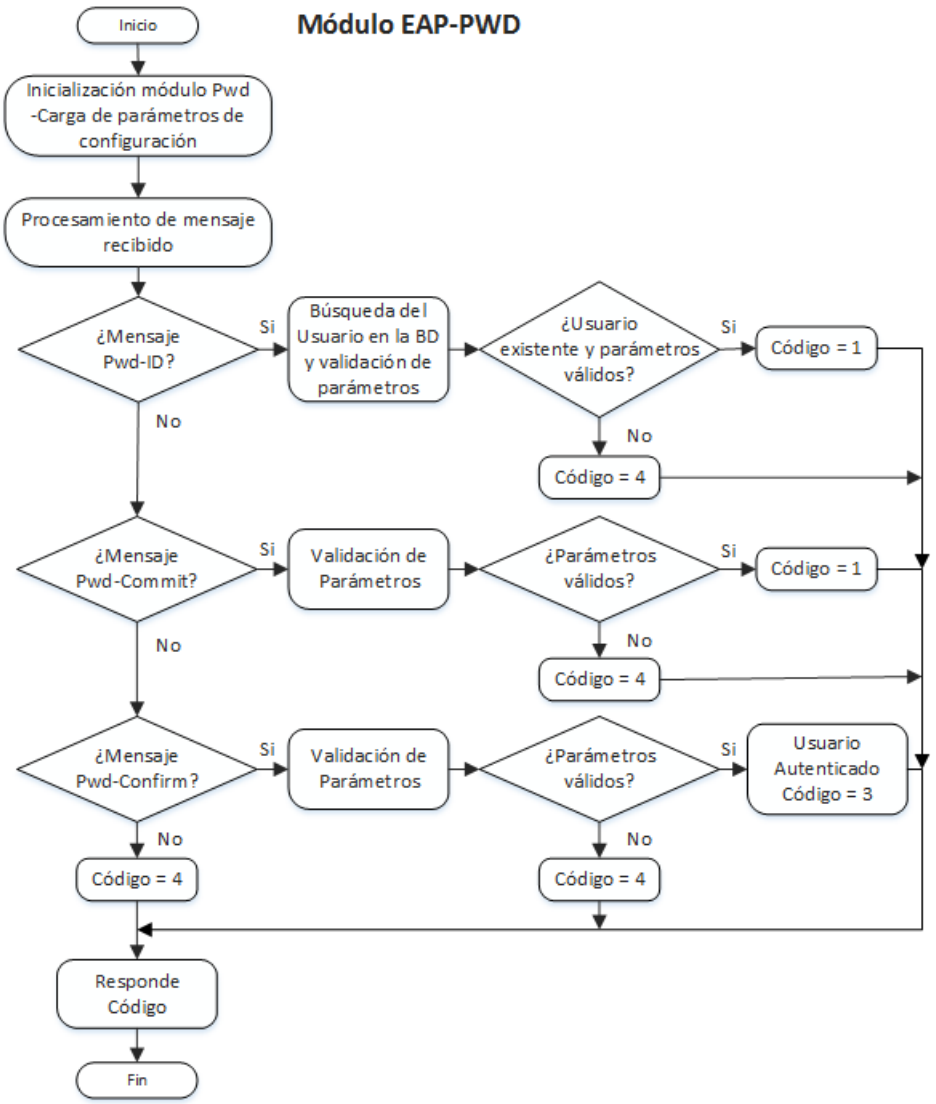

Figura 3.16: Diagrama de flujo del módulo EAP-PWD implementado

Capítulo 4

Estudio y análisis de algunos de los protocolos de autenticación de uso común

Este capítulo tiene como finalidad poner en práctica el uso de las herramientas estudiadas en el capítulo dos para analizar tres protocolos de autenticación diferentes. El primero de ellos se trata del protocolo de Otway-Rees, el cual es una de las primeras propuestas en el área de autenticación. Veremos las bondades que éste presenta y se propondrán algunas mejoras tomando en consideración los trabajos presentados recientemente para el mismo. El segundo protocolo a analizar es el protocolo EAP-PWD, empleado en ambientes Wi-Fi, el cual fue probado con la ayuda de un escenario de pruebas en el capítulo anterior y en este capítulo se realiza su correspondiente análisis formal. Finalmente, se concluye el capítulo con el estudio del protocolo de autenticación EPS-AKA , que se utiliza dentro de sistemas LTE (*Long Term Evolution*). De igual manera, se propone su modelo abstracto y se analiza mediante la herramienta de análisis formal SVO.

4.1. Estudio del protocolo de Otway-Rees

En esta sección se realiza el estudio de un protocolo de autenticación propuesto por Otway y Rees en [33]. A pesar de que se trata de un protocolo publicado en 1987 y de que su uso se haya limitado a pocas aplicaciones, este protocolo cuenta con algunas virtudes que nos ayudarán a entender mejor el uso de las herramientas de análisis de protocolos expuestas en el capítulo dos. En primera instancia, se volverá a exponer el protocolo de Otway-Rees (el cual fue presentado en el capítulo dos) y a continuación se realiza su respectivo análisis

empleando la lógica SVO. En base a los resultados obtenidos a partir del análisis, se discutirán las debilidades y vulnerabilidades que se presentan en este protocolo. Posteriormente, se estudiarán dos de las propuestas de mejora hechas al protocolo de Otway-Rees. La primera de ellas obtenida del artículo de los principios de Abadi y Needham [30], y la segunda de ellas, que es más reciente, fue propuesta por Li Chen en [50]. Para finalizar, se presenta una propuesta de mejora para el protocolo de Otway-Rees elaborada a partir de las mejoras realizadas por Li Chen.

4.1.1. El protocolo de autenticación de Otway-Rees

El protocolo de Otway y Rees es un protocolo de llave simétrica basado en un modelo de tres entidades en el que se cuenta con un centro de distribución de llaves (S) encargado de generar llaves de sesión. Las entidades participantes deben de estar previamente registradas con S y por lo tanto compartir un secreto (una llave) para establecer relaciones de confianza. El protocolo consta de cuatro pasos los cuales se muestran a continuación.

1. $A \rightarrow B: \quad M, ID_a, ID_b, \{N_a, M, ID_a, ID_b\}_{Ka,s}$
2. $B \rightarrow S: \quad M, ID_a, ID_b, \{N_a, M, ID_a, ID_b\}_{Ka,s}, \{N_b, M, ID_a, ID_b\}_{Kb,s}$
3. $S \rightarrow B: \quad M, \{N_a, K_{a,b}\}_{Ka,s}, \{N_b, K_{a,b}\}_{Kb,s}$
4. $B \rightarrow A: \quad M, \{N_a, K_{a,b}\}_{Ka,s}$

En este protocolo, M, y N_a constituyen retos generados por la entidad A. En el mensaje uno, el participante A le hace conocer a el participante B su deseo de establecer una comunicación con él. Le envía un bloque de información cifrada con la llave $K_{a,s}$, la cual el participante B deberá enviar a S junto con información adicional para que S valide a los participantes (mensaje dos). Una vez que S valida que la información recibida es correcta, este genera la llave de sesión $K_{a,b}$ y se la envía a B protegida con las llaves $K_{a,s}$ y $K_{b,s}$ (mensaje tres). Cuando B recibe el mensaje tres, toma la parte cifrada con la llave $K_{b,s}$ y verifica que el *nonce* recibido es el correspondiente al generado para la actual ejecución del protocolo. De ser así, B envía a A la parte cifrada con llave $K_{a,s}$ (mensaje cuatro) para notificarle la llave de sesión generada por S y que ha sido autenticado.

Se puede observar que este protocolo cuenta con una funcionalidad simple y aparentemente eficaz, y que además cumple aparentemente con los objetivos primordiales de todo protocolo de autenticación. Como veremos a continuación, el protocolo de Otway-Rees presenta algunos inconvenientes que se observarán de manera más clara después de realizar el análisis mediante el uso de la lógica SVO.

4.1.2. Análisis formal del protocolo de Otway-Rees

Como se describió en el capítulo dos, el primer paso de todo análisis con la lógica SVO es establecer las premisas iniciales de las que parte el protocolo. Estas son las siguientes:

Premisas Iniciales

P1 $A\ believes(A \overset{Ka,s}{\longleftrightarrow} S)$, $B\ believes(B \overset{Kb,s}{\longleftrightarrow} S)$
P2 $A\ believes\ S\ controls(A \overset{Ka,b}{\longleftrightarrow} B)$, $B\ believes\ S\ controls(A \overset{Ka,b}{\longleftrightarrow} B)$
P3 $A\ believes\ S\ controls(fresh(K_{a,b}))$, $B\ believes\ S\ controls(fresh(K_{a,b}))$
P4 $A\ believes\ fresh(N_a)$, $B\ believes\ fresh(N_b)$
P5 $A\ believes\ A\ has(K_{a,s})$, $B\ believes\ B\ has(K_{b,s})$
P6 $B\ received(\ M, ID_a, ID_b, \{N_a, M, ID_a, ID_b\}_{Ka,s}\ )$
P7 $S\ received(\ M, ID_a, ID_b, \{N_a, M, ID_a, ID_b\}_{Ka,s}, \{N_b, M, ID_a, ID_b\}_{Kb,s}\ )$
P8 $B\ received(\ M, \{N_a, K_{a,b}\}_{Ka,s}, \{N_b, K_{a,b}\}_{Kb,s}\ )$
P9 $A\ received(\ M, \{N_a, K_{a,b}\}_{Ka,s}\ )$
P10 $B\ believes\ B\ received(\ \langle M\rangle_{*B}, ID_a, ID_b, \langle\{N_a, M, ID_a, ID_b\}_{Ka,s}\rangle_{*B}\ )$
P11 $S\ believes\ S\ received(\ \langle M\rangle_{*S}, ID_a, ID_b, \{\langle N_a, M\rangle_{*S}, ID_a, ID_b\}_{Ka,s},$
$\{\langle N_b, M\rangle_{*S}, ID_a, ID_b\}_{Kb,s}\ )$
P12 $B\ believes\ B\ received(\ M, \langle\{N_a, K_{a,b}\}_{Ka,s}\rangle_{*B}, \{N_b, \langle K_{a,b}\rangle_{*B}\}_{Kb,s}\ )$
P13 $A\ believes\ A\ received(\ M, \{N_a, \langle K_{a,b}\rangle_{*A}\}_{Ka,s}\ )$
P14 $A\ believes(\ A\ received\{N_a, \langle K_{a,b}\rangle_{*A}\}_{Ka,s}\ \supset$
$A\ received\{N_a, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A})\}_{Ka,s}\ )$
P15 $B\ believes(\ B\ received\{N_b, \langle K_{a,b}\rangle_{*B}\}_{Kb,s}\ \supset$
$B\ received\{N_b, A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*B})\}_{Kb,s}\ )$

Las premisas uno, dos, tres, cuatro y cinco constituyen las premisas básicas de las que parten la mayoría de los protocolos (suposiciones iniciales). En las premisas seis, siete, ocho y nueve, se transcribe el protocolo a fórmulas para el análisis lógico (premisas de recibimiento de mensajes). Las premisas diez, once, doce y trece, corresponden a las permisas de comprensión de los mensajes. En estas últimas se puede ver que aquellos parámetros desconocidos (de carácter aleatorio) por las entidades se representan mediante la notación $\langle X\rangle_{*P}$ tal como se estableció en el capítulo dos.

Finalmente, las premisas catorce y quince son las premisas de interpretación. Cada entidad asume que si recibe un mensaje en el cual se incluye un valor aleatorio para hacer la función de llave de sesión ($K_{a,b}$), entonces quiere decir que existe una llave de sesión cuyo valor se generó para la ejecución reciente del protocolo.

Gracias a la sencillez del protocolo, no se requiere de ninguna premisa adicional y por lo tanto se procede a hacer la derivaciones lógicas correspondientes a cada entidad.

Derivaciones para el participante A

A1 A *believes* A *received*($\{N_a, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A})\}_{Ka,s}$)

Por P13, Ax7, N, P14, Ax1, MP

A2 A *believes* S *said*($N_a, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A})$)

Por P1, A1, Ax3, N, Ax1, MP

A3 A *believes* S *says*($N_a, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A})$)

Por P4, Ax17, A2, Ax19, N, Ax1, MP

A4 A *believes*$(A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B)$

Por A3, Ax15, P2, N, Ax16, Ax1, MP

A5 A *believes* A *has*$(\langle K_{a,b}\rangle_{*A})$

Por P13, Ax7, P5, Ax8, Ax10, Ax11, N, Ax1, MP

A6 A *believes* $(A \overset{K_{a,b}-}{\longleftrightarrow} B)$ **G3**

Por A4 y A5

A7 A *believes* $fresh(\langle K_{a,b}\rangle_{*A})$ **G5**

Por A3, Ax15, P3, N, Ax16, Ax1, MP

Derivaciones para el participante B

B1 B *believes* B *received*$\{N_b, A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*B})\}_{Kb,s}$

Por P12, Ax7, N, P15, Ax1, MP

B2 B *believes* S *said*($N_b, A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*B})$)

Por P1, B1, Ax3, N, Ax1, MP

B3 B *believes* S *says*($N_b, A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*B})$)

Por P4, Ax17, B2, Ax19, N, Ax1, MP

B4 B *believes*$(A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B)$

Por B3, Ax15, P2, Ax16, N, Ax1, MP

B5 B *believes* B *has*$(\langle K_{a,b}\rangle_{*B})$

Por P12, Ax7, P5, Ax8, Ax10, Ax11, N, Ax1, MP

B6 B *believes* $(A \overset{K_{a,b}-}{\longleftrightarrow} B)$ **G3**

Por B4, B5

B7 B *believes* $fresh(\langle K_{a,b}\rangle_{*B})$ **G5**

Por B3, Ax15, P3, N, Ax16, Ax1, MP

Como se puede apreciar del análisis, el protocolo cumple con el objetivo de la distribución segura de la llave de sesión (metas **G3** y **G5**). Sin embargo, tiene una carencia principal. Esta carencia es, la comprobación que cada entidad realiza para conocer si la otra entidad realmente participó en el intercambio de mensajes del protocolo (metas **G1** y **G2**). La entidad A no puede comprobar si realmente fue B con quien se comunicó, y del mismo modo B tampoco puede saber si A fue quien realmente envió el mensaje uno. Esto da cabida a que un adversario pueda, por ejemplo, capturar el mensaje uno del protocolo e iniciar otra sesión para que B y S consuman una parte de sus recursos atendiendo a la petición. Tanto B como S no tienen manera de verificar que el mensaje recibido por el adversario (haciéndose pasar por A) ya fue recibido previamente, por lo que se podría generar una negación de servicios si el adversario intenta establecer muchas sesiones repitiendo el mismo mensaje en el inicio de la ejecución del protocolo.

Por otro lado, el hecho de que la entidad B atienda y ejecute el resto del procedimiento cada vez que recibe una petición repetida, deja a esta entidad con la creencia de que ha establecido con éxito un proceso de autenticación con la entidad A por cada repetición, siendo esto considerado como un engaño, y por lo tanto un ataque al protocolo (aún en el caso de que el adversario desconozca la llave $K_{a,s}$). De la misma manera, se le puede hacer creer a A que terminó satisfactoriamente la ejecución del protocolo, sin saber si realmente B fue quien le respondió (siendo este último caso más difícil de que ocurra debido a que para ello el adversario tendría que conocer la llave $K_{b,s}$).

4.1.3. Propuesta de mejora del protocolo de Otway-Rees realizada por Abadi y Needham

Como vimos de los resultados anteriores, el protocolo de Otway y Rees presenta algunas vulnerabilidades importantes que hacen que el protocolo no tenga un buen uso práctico. Abadi Y Needham presentaron en su trabajo [30] el protocolo de Otway-Rees como una ilustración del uso de sus principios. En este trabajo resaltaron algunos detalles interesantes que presenta este protocolo como lo es la asociación de identificadores de usuarios con los *nonces* generados por las entidades participantes. Además, Abadi y Needham proponen una modificación al protocolo que es menos costosa en cuestión de cómputo, la cual se presenta a continuación.

1. $A \rightarrow B: \quad ID_a, ID_b, N_a$
2. $B \rightarrow S: \quad ID_a, ID_b, N_a, N_b$
3. $S \rightarrow B: \quad \{N_a, ID_a, ID_b, K_{a,b}\}_{Ka,s}, \{N_b, ID_a, ID_b, K_{a.b}\}_{Kb,s}$
4. $B \rightarrow A: \quad \{N_a, ID_a, ID_b, K_{a,b}\}_{Ka,s}$

Esta propuesta es conceptualmente más simple que la anterior. Se ha reducido la cantidad operaciones de cifrado ya que se libera a A y a B de realizar cifrado en los primeros dos mensajes. Además, se ha eliminado la información redundante presente en la primera propuesta. Sin embargo, se sigue presentando el principal problema de que tanto S como B no tienen manera de saber si A realmente se encuentra activo en el momento de la ejecución del protocolo.

En general, un protocolo de autenticación debe garantizar que ambas entidades realmente han participado en la ejecución del mismo, sin que haya duda alguna de este hecho. Como se mencionó, una repetición del mensaje uno por parte de un adversario podría hacer creer a B que ha completado el proceso de autenticación con el participante A, lo cual se considera como un ataque al protocolo a pesar de que el adversario no tenga forma de obtener la llave de sesión $K_{a,b}$ (ya que puede desconocer la llave $K_{a,s}$). Aunque constituye una ventaja computacional el hecho de que A y B no cifren la información que envían en un inicio, esto abre la posibilidad a otro ataque que se describirá a continuación.

Supóngase que un adversario tiene la posibilidad de capturar, redirigir y modificar los mensajes enviados (y recibidos) por el participante A. Entonces, el adversario puede realizar un ataque en el que recurre a ejecutar el protocolo varias veces en paralelo de acuerdo a su conveniencia como sigue (se emplean diferentes numeraciones para representar a cada una de las ejecuciones del protocolo):

1.		$A \rightarrow E_B:$	ID_a, ID_b, N_a
	I.	$E_B \rightarrow A:$	ID_b, ID_a, N_a
	II.	$A \rightarrow E_S:$	ID_b, ID_a, N_a, N'_a
	i.	$E_A \rightarrow S:$	ID_a, ID_b, N_a, N'_a
	ii.	$S \rightarrow E_A:$	$\{N_a, ID_a, ID_b, K_{a,b}\}_{Ka,s}, \{N'_a, ID_a, ID_b, K_{a,b}\}_{Kb,s}$
2.		$E_B \rightarrow A:$	$\{N_a, ID_a, ID_b, K_{a,b}\}_{Ka,s}$

Aquí, E_A, E_B y E_S representan al mismo adversario suplantando a cada una de las entidades A, B y S respectivamente. Como se puede observar, el adversario recurre a tres ejecuciones casi simultáneas del protocolo. El objetivo del ataque es hacer creer al participante A que ha completado con éxito el proceso de autenticación con B. Para esto, el adversario captura el mensaje de A con destinatario B (mensaje uno de la primera ejecución) de tal forma que B nunca se entera que A desea establecer contacto con él. Inmediatamente después, el adversario procede a iniciar una nueva instancia del protocolo en sentido inverso y envía a A el mensaje I haciéndose pasar por la entidad B. En este mensaje, el adversario añade la información correspondiente al mensaje uno del protocolo y utliza como valor de *nonce* el mismo enviado previamente por A. Cuando A recibe la petición, inicia una nueva sesión

y procede a seguir el paso dos del protocolo comunicándose con S. El adversario captura el mensaje, lo modifica de tal forma que cambia el orden de los identificadores, y realiza una tercera ejecución del protocolo en la cual se hace pasar por la entidad A. En esta tercera ejecución, el adversario se comunica con la entidad S para que genere una llave de sesión que permita la comunicación entre A y B (mensaje i). La entidad S responde enviando la llave de sesión protegida a A (mensaje ii), el adversario captura el mensaje, y le envía a A la parte cifrada con la llave $K_{a,s}$ dentro de la primera ejecución del protocolo. Dado que el *nonce* N_a es el mismo que A generó en un inicio, cuando A reciba el último mensaje y valide N_a, este creerá que finalizó con éxito el proceso de autenticación.

De igual manera, aunque el adversario no tenga forma de conocer la llave de sesión $K_{a,b}$ en este ataque, se propició que la entidad A crea que se encuentra autenticada con B. El hecho de haber reducido la cantidad de operaciones durante la ejecución del protocolo trajo consigo beneficios en cuestión de procesamiento, pero también trajo consecuencias graves para el protocolo ya que el participante A quedó expuesto para ser un oráculo de consultas.

4.1.4. Propuesta de mejora del protocolo de Otway-Rees realizada por Li Chen

En el 2008, Li Chen presentó en [50] una propuesta de mejora del protocolo de Otway-Rees. En su propuesta, Li Chen modifica ligeramente el protocolo con la finalidad de eliminar algunos de los problemas que se presentan en el protocolo de Otway-Rees buscando además que este cumpla con las metas formales de autenticación. El protocolo modificado de Li Chen es el siguiente:

1. $A \rightarrow B: \quad M, ID_a, \{N_a, M, ID_b\}_{Ka,s}$
2. $B \rightarrow S: \quad M, ID_a, ID_b, \{N_a, M, ID_b\}_{Ka,s}, \{N_b, M, ID_a\}_{Kb,s}$
3. $S \rightarrow B: \quad M, \{N_a, ID_b, K_{a,b}\}_{Ka,s}, \{N_b, ID_a, K_{a,b}\}_{Kb,s}$
4. $B \rightarrow A: \quad M, \{N_a, ID_b, K_{a,b}\}_{Ka,s}, \{N_b, ID_b\}_{Ka,b}$
5. $A \rightarrow B: \quad M, \{N_b - 1, ID_a\}_{Ka,b}$

Se puede notar algunos de los cambios realizados en los mensajes, pero destaca principalmente la característica de que se agregó un quinto mensaje en donde A manda una confirmación empleando la llave de sesión recientemente adquirida. Teniendo en cuenta este cambio, más el realizado en el mensaje cuatro en donde B firma una parte del mensaje con la llave de sesión $K_{a,b}$, el análisis con la lógica SVO se presenta a continuación. Como nota, el análisis aquí presentado se realizó de manera independiente al presentado por Li Chen, esto debido a que en su análisis hace uso de algunas interpretaciones a la lógica que difieren ligeramente a lo presentado por Syverson y Oorschot en [28].

Premisas Iniciales

P1 $A\ believes(A \overset{Ka,s}{\longleftrightarrow} S)$, $B\ believes(B \overset{Kb,s}{\longleftrightarrow} S)$

P2 $A\ believes\ S\ controls(A \overset{Ka,b}{\longleftrightarrow} B)$, $B\ believes\ S\ controls(A \overset{Ka,b}{\longleftrightarrow} B)$

P3 $A\ believes\ S\ controls(fresh(K_{a,b}))$, $B\ believes\ S\ controls(fresh(K_{a,b}))$

P4 $A\ believes\ fresh(N_a)$, $B\ believes\ fresh(N_b)$

P5 $A\ believes\ A\ has(K_{a,s})$, $B\ believes\ B\ has(K_{b,s})$

P6 $B\ believes\ B\ received(\ M, ID_a, \langle\{N_a, M, ID_b\}_{Ka,s}\rangle_{*B}\ )$

P7 $S\ believes\ S\ received(\ M, ID_a, ID_b, \{\langle N_a\rangle_{*S}, M, ID_b\}_{Ka,s}, \{\langle N_b\rangle_{*S}, M, ID_a\}_{Kb,s}\ )$

P8 $B\ believes\ B\ received(\ M, \langle\{N_a, ID_b, K_{a,b}\}_{Ka,s}\rangle_{*B}, \{N_b, ID_a, \langle K_{a,b}\rangle_{*B}\}_{Kb.s}\ )$

P9 $A\ believes\ A\ received(\ M, \{N_a, ID_b, \langle K_{a,b}\rangle_{*A}\}_{Ka,s}, \{\langle N_b\rangle_{*A}, ID_b\}_{\langle Ka,b\rangle *A}\ )$

P10 $B\ believes\ B\ received(\ M, \{N_b - 1, ID_a\}_{Ka,b}\ )$

P11 $A\ believes(\ A\ received\{N_a, ID_b, \langle K_{a,b}\rangle_{*A}\}_{Ka,s}\ \supset$
$A\ received\{N_a, ID_b, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A})\}_{Ka,s}\ )$

P12 $B\ believes(\ B\ received\{N_b, ID_a, \langle K_{a,b}\rangle_{*B}\}_{Kb,s}\ \supset$
$B\ received\{N_b, ID_a, A \overset{\langle K_{a,b}\rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*B})\}_{Kb,s}\ )$

P13 $A\ believes(\ fresh(\langle K_{a,b}\rangle_{*A})\ \wedge\ A\ received\{\langle N_b\rangle_{*A}, ID_b\}_{\langle Ka,b\rangle *A}\ \supset$
$fresh(\langle N_b\rangle_{*A}, ID_b)\ )$

Derivaciones para el participante A

A1 $A\ believes\ A\ received(\ \{N_a, ID_b, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A})\}_{Ka,s}\ )$
Por P9, Ax7, P11, N, Ax1, MP

A2 $A\ believes\ S\ said(\ N_a, ID_b, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A})\ )$
Por A1, P1, N, Ax3, Ax1, MP

A3 $A\ believes\ S\ says(\ N_a, ID_b, A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B, fresh(\langle K_{a,b}\rangle_{*A})\ )$
Por A2, P4, Ax17, Ax19, N, Ax1, MP

A4 $A\ believes(A \overset{\langle K_{a,b}\rangle_{*A}}{\longleftrightarrow} B)$
Por A3, Ax15, P2, N, Ax16, Ax1, MP

A5 $A\ believes\ A\ has(\langle K_{a,b}\rangle_{*A})$
Por P9, Ax7, P5, Ax8, Ax10, Ax11, N, Ax1, MP

A6 $A\ believes\ (A \overset{K_{a,b^-}}{\longleftrightarrow} B)$ **G3**
Por A4 y A5

A7 $A\ believes\ fresh(\langle K_{a,b}\rangle_{*A})$ **G5**
Por A3, Ax15, P3, N, Ax16, Ax1, MP

A8 $A\ believes\ A\ received(\ \{\langle N_b\rangle_{*A}, ID_b\}_{\langle Ka,b\rangle *A}\ )$
Por P9, N, Ax7, Ax1, MP

A9 A *believes* B *said*($\langle N_b \rangle_{*A}, ID_b$)
Por A4, A8, N, Ax3, Ax1, MP

A10 A *believes* B *says*($\langle N_b \rangle_{*A}, ID_b$) **G1**
Por A7, A8, P13, A9, N, Ax19, Ax1, MP

A11 A *believes*($A \overset{K_{a,b}-}{\longleftrightarrow} B \ \wedge \ B$ *says*$\{\langle N_b \rangle_{*A}, ID_b\}_{\langle Ka,b \rangle *A}$) **G4**
Por A6, A10

Derivaciones para el participante B

B1 B *believes* B *received*$\{N_b, ID_a, A \overset{\langle K_{a,b} \rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b} \rangle_{*B})\}_{Kb,s}$
Por P8, N, Ax7, P12, Ax1, MP

B2 B *believes* S *said*($N_b, ID_a, A \overset{\langle K_{a,b} \rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b} \rangle_{*B})$)
Por P1, B1, N, Ax3, Ax1, MP

B3 B *believes* S *says*($N_b, ID_a, A \overset{\langle K_{a,b} \rangle_{*B}}{\longleftrightarrow} B, fresh(\langle K_{a,b} \rangle_{*B})$)
Por P4, Ax17, B2, N, Ax19, Ax1, MP

B4 B *believes*$(A \overset{\langle K_{a,b} \rangle_{*B}}{\longleftrightarrow} B)$
Por B3, Ax15, P2, N, Ax16, Ax1, MP

B5 B *believes* B *has*$(\langle K_{a,b} \rangle_{*B})$
Por P8, Ax7, P5, Ax8, Ax10, Ax11, N, Ax1, MP

B6 B *believes* $(A \overset{K_{a,b}-}{\longleftrightarrow} B)$ **G3**
Por B4, B5

B7 B *believes* $fresh(\langle K_{a,b} \rangle_{*B})$ **G5**
Por B3, Ax15, P3, N, Ax16, Ax1, MP

B8 B *believes* B *received*$\{N_b - 1, ID_a\}_{\langle Ka,b \rangle *B}$
Por P10, N, Ax7, Ax1, MP

B9 B *believes* A *said*($N_b - 1, ID_a$)
Por B4, B8, N, Ax3, Ax1, MP

B10 B *believes* A *says*($N_b - 1, ID_a$) **G1, G2**
Por P4, Ax17, B9, N, Ax19, Ax1, MP

B11 B *believes*($A \overset{K_{a,b}-}{\longleftrightarrow} B \ \wedge \ A$ *says*$\{N_b - 1, ID_a\}_{\langle Ka,b \rangle *B}$) **G4**
Por B6, B10

Del análisis presentado podemos concluir algunas cosas. En primer lugar, debido a que el participante A confirma tener conocimiento de la llave de sesión $K_{a,b}$, ya es posible demostrar formalmente que el protocolo cumple con la mayoría de las metas requeridas por todo protocolo de autenticación, lo que hace que este protocolo pueda ser considerado como seguro. En segundo lugar, si bien no es posible para A cumplir la meta **G2**, esto no implica que sea

un punto vulnerable al protocolo ya que el participante B se autentica ante A demostrándole tener conocimiento de la llave $K_{a,b}$ (meta **G4**). En general, se puede afirmar que si un protocolo de autenticación incluye un intercambio de llaves, el proceso de demostrar el conocimiento de la llave recién generada por parte de ambas entidades incluirá la presentación de ciertos datos, los cuales pueden ser utilizados adicionalmente como información para llevar a cabo el proceso de autenticación. Para el caso de la entidad B, si es posible cumplir con la meta **G2**, esto debido a que dicha meta especifica que A debe enviar a B un mensaje que incluya una función del nonce N_b, lo cual ocurre dentro del mensaje cinco. Se puede notar que la derivación de la meta **G2** para B constituye como tal un paso para la derivación de la meta **G4**, demostrando que hay un enlace entre ambas metas (autenticación y confirmación de llave) y haciendo así válida la afirmación recientemente dicha. Por lo tanto, A no requiere cumplir estrictamente con la meta **G2**.

Finalmente, se puede apreciar que dentro del protocolo existen ciertos datos y operaciones que no tienen repercusión ni en el funcionamiento del mismo ni en el análisis formal, por lo que terminan siendo redundantes. En la siguiente sección hablaremos más acerca de estos detalles y se proponen las alternativas necesarias para corregirlos en base al uso de los principios de diseño presentados en el capítulo dos.

4.1.5. Consideraciones adicionales y propuesta de mejora del protocolo de Otway-Rees

Existen algunos detalles que se presentan en la propuesta de Li Chen que pueden ser mejorados. Dentro de estos detalles se encuentran algunos relacionados al manejo de información redundante en el mismo y otros son más relacionados con la funcionalidad. Comenzaremos hablando acerca de los detalles relacionados al manejo de información redundante. Para ello nos basaremos en las ideas expuestas por lo principios de Abadi y Needham. Las consideraciones adicionales son las siguientes:

- Como se había comentado desde la versión original del protocolo Otway-Rees, el uso del nonce M resulta innecesario. Eliminar este nonce de los mensajes ayuda a tener un menor procesamiento sobre todo en aquellas partes en donde se cifra junto con otros datos (mensaje uno y dos). De acuerdo al principio cuatro, las operaciones de cifrado son costosas en cuestiones de cómputo, por lo que no preguntarse para que se realiza y que datos son los que se cifran constituye un error de diseño.

- Con respecto al mensaje tres, no es necesario el cifrado de los identificadores de A y B dentro del mensaje ($\{N_a, ID_b, K_{a,b}\}_{Ka,s}, \{N_b, ID_a, K_{a,b}\}_{Kb,s}$). Cada entidad conserva

un registro (durante la ejecución del protocolo) de con quien ha utilizado un valor de nonce específico. Por lo que tanto el nonce N_a como el nonce N_b sirven como una referencia segura hacia cada entidad así como al destinatario. El hecho de cifrar desde un inicio N_a y N_b garantiza una referencia unívoca y segura para cada participante tal como se discutió en el principio seis (ver capítulo dos).

- En el mensaje cuatro, se tiene que el participante B envía una prueba de que conoce la llave $K_{a,b}$ a la entidad A. Dentro de este mensaje, se encuentra el nonce N_b que servirá como reto, y el identificador de B. Agregar este identificador dentro del mensaje es totalmente necesario para que la entidad A pueda tener la certeza de que el nonce no haya sido alterado en el camino. Relacionando el mensaje cuatro con el mensaje cinco, se puede ver que A realiza el mismo proceso que B, pero realiza una operación (de sustracción) sobre N_b para demostrarle a B que lo recibió y que se encuentra en posesión de $K_{a,b}$. Prestando atención a los detalles, se sabe que la entidad B tiene conocimiento del valor de N_b, por lo que el mismo puede darse cuenta si el mensaje cinco ha sido alterado sin la necesidad de información adicional. De esto se puede concluir que el identificador de A (ID_a) es redundante en el mensaje y puede omitirse. Por otro lado, el hecho de extraer N_b del mensaje firmado con $K_{a,b}$ en el mensaje cuatro y luego volverlo a cifrar con la llave $K_{a,b}$ para el mensaje cinco constituye como tal una operación realizada al mismo. Realizar la operación de resta está de más y puede ser omitida.

Teniendo en cuenta estos detalles, se puede presentar una propuesta mas compacta y mejorada del protocolo, la cual puede ser la siguiente:

1. $A \rightarrow B: \quad ID_a, \{N_a, ID_b\}_{Ka,s}$
2. $B \rightarrow S: \quad ID_a, ID_b, \{N_a, ID_b\}_{Ka,s}, \{N_b, ID_a\}_{Kb,s}$
3. $S \rightarrow B: \quad \{N_a, K_{a,b}\}_{Ka,s}, \{N_b, K_{a,b}\}_{Kb.s}$
4. $B \rightarrow A: \quad \{N_a, K_{a,b}\}_{Ka,s}, \{N_b, ID_b\}_{Ka,b}$
5. $A \rightarrow B: \quad \{N_b\}_{Ka,b}$

Se puede observar que esta versión sigue conservando las propiedades vistas en la propuesta de Li Chen, por lo que un análisis SVO nos conduciría al mismo resultado expuesto en el apartado anterior. La diferencia se encuentra en que se ha eliminado todo lo redundante del protocolo, haciendo que este posea un mayor rendimiento. Como se ha venido mencionado, las operaciones de cifrado son operaciones costosas por lo que entre menos información sea cifrada, se liberará más carga a las entidades involucradas.

A pesar de resolver el principal problema de seguridad en el protocolo de Otway-Rees al añadirle un quinto mensaje de confirmación, y de haber pulido el protocolo gracias al uso

de los principios de diseño de Abadi y Needham, en el aspecto funcional de este protocolo se siguen presentando algunos incovenientes. Supongamos que un adversario realiza muchas peticiones concurrentes (o casi simultáneas) de autenticación a la entidad B. B continúa de acuerdo al protocolo y atiende cada una de las peticiones recibidas enviando la información recibida de cada petición más información adicional al centro de distribución de llaves S. S comprueba que los identificadores de usuarios sin cifrar coincidan con los que se encuentran cifrados y de ser así, genera las llaves de sesión necesarias para atender cada solicitud de B. B entonces recibe los mensajes de S, realiza un par de operaciones (descifrado y cifrado de mensajes) y envía el mensaje cuatro en espera de una respuesta. De esto podemos observar dos cosas:

- En primer lugar, la entidad S genera y emite llaves de sesión sin un control estricto, esto es, sólo valida que los identificadores de usuario coincidan para generar una llave. Esto le permite a un adversario que ha capturado el mensaje uno de alguna ejecución previa realizada por A (o cualquier otra entidad registrada con S) poder repetir dicho mensaje cuantas veces desee para hacer que S genere llaves de sesión sin que este note que se trata del mismo mensaje.

- Por otra parte, en la misma medida que S gasta recursos generando y protegiendo las llaves de sesión construidas para una misma petición, B también consume recursos atendiendo a una misma petición repetida. Entre más peticiones repetidas realice el adversario, mayor será el número de recursos que ocupen B y S por lo que se puede propiciar que estas entidades colapsen, generando un ataque de denegación de servicios. Aquellas entidades legítimas que deseen establecer contacto con B o con S no podrán recibir los servicios de B o S por la falta de recursos disponibles en los mismos.

Es común que los diseñadores de protocolos de autenticación no se enfoquen en este tipo de vulnerabilidades. Pero de acuerdo a lo anterior resulta claro que algo debe hacerse para corregir estos inconvenientes. Una manera sencilla de solucionar esto es haciendo que S lleve un registro de los valores de *nonce* que han sido empleados en ejecuciones previas del protocolo. Así, cuando el adversario envíe una repetición de un mensaje previo, S podrá notar que el *nonce* recibido de la repetición ya ha sido utilizado y podrá por lo tanto descartar la petición. Si bien esta esta es una alternativa simple para resolver el problema, también presenta un par de inconvenientes: el primero es el espacio de memoria requerido para almacenar todos los valores que se reciben; y el segundo es la carga adicional que se genera durante el proceso de comparación de los valores de *nonce* recibidos. Por lo tanto esta no resulta ser una opción viable. Dos alternativas a este problema se presentan a continuación.

La primera alternativa consiste en el uso de estampas de tiempo dentro del protocolo. Suponiendo que las entidades involucradas poseen sincronía de relojes, la entidad A podría utilizar una estampa de tiempo T_a y enviarla en el mensaje uno en lugar del *nonce* N_a. El protocolo queda de la siguiente manera:

1. $A \rightarrow B: \quad ID_a, \{T_a, ID_b\}_{Ka,s}$
2. $B \rightarrow S: \quad ID_a, ID_b, \{T_a, ID_b\}_{Ka,s}, \{N_b, ID_a\}_{Kb,s}$
3. $S \rightarrow B: \quad \{T_a, K_{a,b}\}_{Ka,s}, \{N_b, K_{a,b}\}_{Kb,s}$
4. $B \rightarrow A: \quad \{T_a, K_{a,b}\}_{Ka,s}, \{N_b, ID_b\}_{Ka,b}$
5. $A \rightarrow B: \quad \{N_b\}_{Ka,b}$

Aquí, la estampa de tiempo T_a cumple además con la función de nonce de A. S realiza la siguiente validación cuando recibe el mensaje dos y descifra la parte correspondiente a la entidad A:

$$T_s - \triangle \leq T_a \leq T_s + \triangle$$

En donde T_s es el tiempo local del servidor y $\triangle$ es una ventana de tiempo preestablecida. Cuando S recibe T_a, revisa que esta estampa se encuentre dentro de una ventana de tiempo aceptable de acuerdo a su conocimiento del tiempo, y de ser así, toma como válida la petición para generar la llave $K_{a,b}$. En el caso contrario, S invalida la petición y le puede notificar a B que la solicitud no fue aceptada. Con esto, el adversario queda con menores posibilidades de poder efectuar repeticiones de los mensajes, ya que las solicitudes realizadas con los mismos serán rechazadas después de un tiempo establecido por la ventana. Entre mayor sea la ventana de tiempo, mayores posibilidades tendrá el adversario para poder realizar ataques de repetición.

Otra característica interesante de emplear esta estrategia es que la cantidad de operaciones que deben realizar B y S queda reducida por el hecho de que ambas entidades terminan la ejecución del protocolo desde el momento en que S nota que el mensaje está fuera de tiempo. Por lo tanto, S puede ahorrarse la construcción del mensaje tres y B la del mensaje cuatro del protocolo, disminuyendo así la cantidad de operaciones de cifrado a realizar de manera innecesaria.

Si bien esta estrategia ayuda a aligerar la carga de S y B cuando un adversario intenta replicar mensajes caducados, se siguen presentando los inconvenientes comunes de todo protocolo de autenticación que se basa en el uso de estampas de tiempo para demostrar la frescura de los mensajes. En primera instancia, se requiere de un sistema seguro de sincronía de relojes, lo cual genera un punto más de vulnerabilidad al mecanismo de autenticación debido a que un adversario podría optar por empezar atacando el sistema de sincronía antes

que el protocolo de autenticación. En segundo lugar, establecer el valor adecuado para la ventana de tiempo depende de varios factores de carácter aleatorio como lo es el retardo de los mensajes debido al tráfico de la red.

Una segunda alternativa es el uso de valores de *nonce* que puedan ser validados por S sin la necesidad de hacer una búsqueda exhaustiva dentro de sus registros. Para lograr esto, se puede proponer una estrategia en la que se hace uso de cantidades predecibles para S, como lo son los valores generados por un contador. De acuerdo con lo mencionado por el principio siete, toda cantidad predecible puede ser utilizada como garantía de frescura siempre y cuando se tomen las medidas necesarias para protegerla. Por lo tanto, se puede sustituir el valor del *nonce* N_a por un valor de contador C_a tal como se hizo en la alternativa anterior con T_a, quedando el protocolo como sigue:

1. $A \rightarrow B: \quad ID_a, \{C_a, ID_b\}_{Ka,s}$
2. $B \rightarrow S: \quad ID_a, ID_b, \{C_a, ID_b\}_{Ka,s}, \{N_b, ID_a\}_{Kb,s}$
3. $S \rightarrow B: \quad \{C_a, K_{a,b}\}_{Ka,s}, \{N_b, K_{a,b}\}_{Kb,s}$
4. $B \rightarrow A: \quad \{C_a, K_{a,b}\}_{Ka,s}, \{N_b, ID_b\}_{Ka,b}$
5. $A \rightarrow B: \quad \{N_b\}_{Ka,b}$

En esta propuesta se puede observar que sólo la entidad A utiliza un contador como valor de *nonce*, mientras que B se mantiene utilizando un valor de *nonce* aleatorio. A protege el valor de este contador cifrándolo con la llave $K_{a,s}$, por lo que no será visible para el adversario. Tanto A como S deben de llevar un registro del contador, estableciendo desde el principio su valor de inicio y manteniendo hasta donde sea posible una sincronía del mismo. La validación realizada por S es:

$$C_s \leq C_a \leq C_s + \triangle$$

Donde Cs es el valor del contador almacenado por S y $\triangle$ es una ventana de valores por encima del valor actual permitidos. Cada vez que la entidad A emite en el mensaje uno el valor actual de su contador, este lo incrementa. De esta manera, cada vez que la entidad A inicie el proceso de autenticación, lo hará utilizando un valor diferente del contador. Por su parte, cuando S recibe el mensaje dos, obtiene el contador C_a y lo verifica con respecto al suyo. Si C_a es igual (lo cual es lo ideal) o se encuentra dentro de un rango de valores mayores a C_s definidos por la ventana, la petición será considerada válida y S incrementa de la misma manera que A su contador continuando además con el procedimiento. Si C_a es menor que el valor del contador C_s, este descarta la petición debido a que ese valor ya fue utilizado con anterioridad. De igual manera, si C_a se encuentra por arriba del valor C_s más la ventana, la petición será descartada pero esta vez debido a que C_a se encuentra muy adelantado y

se ha perdido la sincronía. En cualesquiera de los casos en donde la validación falla, S no incrementa el valor de C_s.

El caso de la pérdida de sincronía puede ocurrir cuando la entidad A ha iniciado varios procesos de autenticación en el pasado y no los concluyó por alguna razón ajena a A. Como A actualiza el valor de su contador cada vez que inicia un proceso de autenticación, si su petición no llega a ser enviada a S, inevitablemente la sincronía del contador se perderá. Una vez perdida la sincronía, la entidad A no podrá autenticarse, por lo que es conveniente iniciar algún proceso de sincronización del contador.

La medida a la que se recurre para que A no pierda la sincronía con solo una falla en la ejecución del protocolo es utilizar la ventana. Con ella, se tendrá un rango de valores aceptables para S en los cuales no se recurrirá al proceso de sincronía. La estrategia a seguir para realizar el proceso de sincronía del contador puede ser implementada de varias maneras, pero esto queda fuera de los objetivos de este trabajo.

Otro aspecto que resulta importante mencionar es acerca de la longitud (en bits, bytes, etc.) que debe tener el contador. Sólo podemos decir que como medida de seguridad, la longitud del contador debe ser seleccionada de tal manera que sea prácticamente imposible su desbordamiento. Con esto es posible garantizar que cada petición realizada por A será única. Un adversario que tenga la posibilidad de capturar y suprimir algún mensaje de inicio del protocolo por parte de la entidad A, sólo tendrá la oportunidad de replicarlo una vez, por lo que no podrá hacer que S realice tantas operaciones innecesarias con la misma petición.

De acuerdo los argumentos discutidos, se pudo observar que el uso de las estampas de tiempo o de los contadores dentro del protocolo de Otway- Rees como garantías de frescura permiten hacerle frente a los ataques por denegación de servicios de una manera efectiva. Estas estrategias ayudan a reducir el costo de las operaciones realizadas por las entidades afectadas, y además proporcionan una manera de evitar que el centro de distribución de llaves genere llaves de sesión sin tener control alguno. De las dos estrategias propuestas para resolver este problema, se puede decir que la segunda cuenta con una ventaja adicional, que es, su sencillez. Mientras el uso de estampas de tiempo depende de todo un sistema de sincronización del tiempo que actúe de forma regular (enviando mensajes de actualización y generando así una sobrecarga adicional al sistema), la utilización de contadores sólo necesita de una sincronización inicial. En ambientes distribuidos, en donde sea complicado mantener una base del tiempo, la propuesta de los contadores resulta en una muy buena estrategia para tener un sistema de autenticación confiable y eficaz.

Finalmente, en cuestión del análisis formal con el uso de la lógica SVO, podemos decir que los cambios expuestos en las estrategias recién mencionadas no afectan al análisis formal que se realizó para la propuesta de Li Chen, ya que como tal, no modifican el objetivo

funcional de ningún parámetro involucrado en la demostración de las metas formales, siendo así innecesario elaborar otro análisis. Este hecho nos indica que la lógica SVO no cuenta con la expresividad suficiente para ayudarnos a establecer si un protocolo es vulnerable a un ataque por denegación de servicios. Por lo tanto, es necesario recurrir a estrategias informales para determinar si un protocolo puede lidiar con este problema.

4.2. Protocolos de autenticación en ambientes inalámbricos

En esta sección se estudiarán dos protocolos de autenticación empleados en estándares modernos de comunicaciones inalámbricas. El primero de ellos es el protocolo EAP-PWD, empleado dentro de los sistemas 802.11 (Wi-Fi), mientras que el segundo se trata del protocolo denominado EPS-AKA, empleado en LTE.

4.2.1. Acceso y autenticación en sistemas Wi-Fi

Como se vio en el capítulo tres, el acceso en ambientes inalámbricos para el estándar 802.11 se puede establecer mediante el uso del Protocolo de Autenticación Extensible (EAP por sus siglas en inglés). El protocolo EAP es un *framework* que encapsula una variedad de métodos de autenticación los cuales se utilizan entre una entidad cliente y un servidor de autenticación. Existen en la actualidad una gran diversidad de métodos de autenticación soportados por EAP, pero tal como se mencionó, sólo algunos de ellos son de uso común entre los fabricantes de dispositivos que siguen el estándar EAP.

En éste apartado sólo nos limitaremos a analizar el método de autenticación EAP-PWD (EAP-Password). La elección de éste método se debe a que su evaluación formal no ha sido presentada en trabajos anteriores sobre análisis de protocolos criptográficos. La evaluación formal mediante la lógica SVO de otros métodos como EAP-TLS, EAP-TTLS y EAP-AKA fue propuesta en [19].

Tal como se especifica en el RFC 5931 [48], y atendiendo a lo expuesto en el capítulo anterior, el método de autenticación EAP-PWD consiste del siguiente intercambio de mensajes:

1. *EAP-pwd-ID/Request* ($Servidor\ EAP \rightarrow Par\ EAP$)
   Ciphersuite, Token, Password Processing Method, Server_ID
2. *EAP-pwd-ID/Response* ($Par\ EAP \rightarrow Servidor\ EAP$)
   Ciphersuite, Token, Password Processing Method, Peer_ID

3. *EAP-pwd-Commit/Request* (*Servidor EAP* → *Par EAP*)
   Scalar_S, Element_S
4. *EAP-pwd-Commit/Response* (*Par EAP* → *Servidor EAP*)
   Scalar_P, Element_P
5. *EAP-pwd-Confirm/Request* (*Servidor EAP* → *Par EAP*)
   Confirm_S
6. *EAP-pwd-Confirm/Response* (*Par EAP* → *Servidor EAP*)
   Confirm_P

Como se describió en el capítulo tres, el intercambio *EAP-pwd-ID* tiene como objetivo el descubrimiento de la identidad de cada entidad (*Server_ID*, *Peer_ID*) así como la negociación de parámetros para la técnica criptográfica a emplear (*Ciphersuite*, *Token*, *Password Processing Method*). Se tienen dos técnicas criptográficas disponibles para el enmascaramiento de la contraseña, una basada en el uso de la Criptografía de Campos Finitos (FFC por sus siglas en inglés) y la otra basada en el uso de la Criptografía de Curvas Elípticas (ECC por sus siglas en inglés). Las contraseña utilizada por un usuario (*Par EAP*) puede ser débil (de baja entropía).

En el intercambio *EAP-pwd-Commit*, las entidades participantes intercambian información para generar un secreto compartido (*Scalar_S*, *Element_S*, *Scalar_P*, *Element_P*). Los valores *"Scalar"* están formados a partir de una suma de dos números aleatorios conocidos como *s_rand*, *s_mask* para el *Servidor EAP* y *p_rand*, *p_mask* para el *Par EAP*. Los valores denominados *"Element"* se construyen a partir de una función (específica de cada técnica criptográfica) que tiene como parámetros a *s_mask* y la contraseña para el caso del *Servidor EAP*, y *p_mask* y la contraseña para el caso del *Par EAP*. Dicha función constituye el inverso de un número dentro de FFC o ECC, lo cual involucra el problema del logaritmo discreto.

Finalmente, en el intercambio *EAP-pwd-Confirm*, el *Par EAP* y el *Servidor EAP* prueban que se encuentran activos y que poseen conocimiento de la contraseña y de la llave de sesión generada para la ejecución del protocolo en cuestión. Los valores *"Confirm"* están formados a partir de una función HMAC-SHA256 (ver RFC 4634 [51]) que toma como parámetros *Scalar_S*, *Element_S*, *Scalar_P* y *Element_P* y la llave de sesión principalmente. Estos elementos se concatenan y se introducen a la función HMAC ordenados de diferente manera para cada entidad con la finalidad de que *Confirm_S* y *Confirm_P* contengan un valor distinto. La llave de sesión se obtiene a partir de una función que involucra a los elementos *s_rand*, *Scalar_P*, *Element_P*, *p_rand*, *Scalar_S*, *Element_S* y la contraseña. Para mayor aclaración se puede consultar [48].

Habiendo explicado los principios y el procedimiento de autenticación de usuarios en base al protocolo EAP-PWD, ahora es posible obtener una idea abstracta del protocolo para realizar el análisis formal del mismo. Esto se describirá a continuación.

4.2.1.1. Análisis formal del protocolo EAP-PWD

El primer paso para analizar el protocolo EAP-PWD con la lógica SVO es obtener una idea abstracta de este. Para realizar esta labor, es necesario tener una idea clara de las operaciones que se realizan dentro del mismo, así como entender el objetivo de cada parámetro enviado en el intercambio de mensajes. Como es común en todo protocolo de comunicaciones, algunos de los datos enviados en los mensajes tienen carácter informativo, pudiéndose así omitir este tipo de información.

De acuerdo a la descripción expuesta, y modelando a el Par EAP y al Servidor EAP como las entidades A y B respectivamente, el protocolo EAP_PWD realiza lo siguiente:

1. $A \rightarrow B: \quad ID_a$
2. $B \rightarrow A: \quad N_b, F(K_{a,b}, Y_b)$
3. $A \rightarrow B: \quad N_a, F(K_{a,b}, Y_a)$
4. $B \rightarrow A: \quad \{H(F(K_{a,b}, Y_b), N_b, F(K_{a,b}, Y_a), N_a)\}_{Ks}$
5. $A \rightarrow B: \quad \{H(F(K_{a,b}, Y_a), N_a, F(K_{a,b}, Y_b), N_b)\}_{Ks}$

Aquí, N_a representa a $Scalar\_P$ y y N_b representa a $Scalar\_S$. Como se mencionó, $Scalar\_P$ y $Scalar\_S$ se forman a partir de una suma de dos números aleatorios (p_rand+p_mask en el primer caso, y s_rand+s_mask en el segundo), por lo que claramente pueden ser interpretados como *nonces*. $F(K_{a,b}, Y_a)$ y $F(K_{a,b}, Y_b)$ son los parámetros $Element\_P$ y $Element\_S$ respectivamente. Y_a y Y_b constituyen los parámetros p_mask y s_mask, lo cuales realizan un enmascaramiento a la contraseña compartida entre ambas entidades. Dicha contraseña puede ser vista como una llave compartida, de ahí que su representación sea $K_{a,b}$. La función F depende de la técnica criptográfica elegida (FFC o ECC), pero para ambos casos requiere de los mismos parámetros.

La función H representa a algoritmo HMAC-SHA256 que se emplea dentro del protocolo para el cálculo de $Confirm\_S$ y $Confirm\_P$. Debido a que dentro de los parámetros ingresados a esta función se encuentra la llave de sesión generada, se puede interpretar este hecho como si el resultado de H fuera cifrado con dicha llave. K_s es la llave de sesión generada durante una ejecución particular del protocolo. Esta llave es función de algunos parámetros intercambiados previamente, entre los cuales se tienen p_mask, s_mask, $Scalar\_P$, $Scalar\_S$ y la contraseña, por lo que puede representarse como $K_s = F(Y_a, Y_b, N_a, N_b, K_{a,b})$.

Se puede ver que el modelo abstracto propuesto en este trabajo no incluye la negociación de los parámetros de la técnica criptográfica a emplear (intercambio *EAP-pwd-ID*). Dicha negociación como tal no forma parte de la funcionalidad de autenticación e intercambio de llaves del protocolo además de que no aporta datos relevantes al análisis formal. Esto se debe a que la información enviada durante esa etapa está en claro (sin cifrar). En su lugar, se optó por sólo poner un mensaje en el que A le indica a B su deseo iniciar un proceso de autenticación. El análisis formal mediante el uso de la lógica SVO para el protocolo EAP-PWD es el siguiente.

Premisas iniciales

P1 $A\ believes\ A \overset{Ka,b}{\longleftrightarrow} B$, $B\ believes\ A \overset{Ka,b}{\longleftrightarrow} B$

P2 $A\ believes\ A\ has(K_{a,b})$, $B\ believes\ B\ has(K_{a,b})$

P3 $A\ believes\ A\ has(N_a)$, $B\ believes\ B\ has(N_b)$

P4 $A\ believes\ fresh(N_a)$, $B\ believes\ fresh(N_b)$

P5 $A\ believes\ A\ has(Y_a)$, $B\ believes\ B\ has(Y_b)$

P6 $A\ believes\ fresh(Y_a)$, $B\ believes\ fresh(Y_b)$

P7 $A\ believes(\ N_a, Y_a\ )$, $B\ believes(\ N_b, Y_b\ )$

P8 $B\ believes\ B\ received(\ ID_a\ )$

P9 $A\ believes\ A\ received(\ \langle N_b\rangle_{*A}, \langle F(K_{a,b}, Y_b)\rangle_{*A}\ )$

P10 $B\ believes\ B\ received(\ \langle N_a\rangle_{*B}, \langle F(K_{a,b}, Y_a)\rangle_{*B}\ )$

P11 $A\ believes\ A\ received(\ \{H(F(K_{a,b}, Y_b), N_b, F(K_{a,b}, Y_a), N_a)\}_{Ks}\ )$

P12 $B\ believes\ B\ received(\ \{H(F(K_{a,b}, Y_a), N_a, F(K_{a,b}, Y_b), N_b)\}_{Ks}\ )$

P13 $A\ believes(\ (A\ has(F(K_{a,b}, Y_a))\ \wedge\ A\ has(\langle F(K_{a,b}, Y_b)\rangle_{*A})\ \wedge\ A\ has(N_a)\ \wedge$
$A\ has(\langle N_b\rangle_{*A})\ \wedge\ A\ has(K_{a,b}))\ \supset\ A\ has(K_s)\ ),$
$B\ believes(\ (B\ has(\langle F(K_{a,b}, Y_a)\rangle_{*B})\ \wedge\ B\ has(\langle F(K_{a,b}, Y_b))\ \wedge\ B\ has(\langle N_a\rangle_{*B})\ \wedge$
$B\ has(N_b)\ \wedge\ B\ has(K_{a,b}))\ \supset\ B\ has(K_s)\ )$

P14 $A\ believes(\ A\ received(H(F(K_{a,b}, Y_b), N_b, F(K_{a,b}, Y_a), N_a))\ \wedge\ A\ believes(N_a, Y_a)\ \wedge$
$A \overset{Ka,b}{\longleftrightarrow} B\ \supset\ A \overset{Ks}{\longleftrightarrow} B\ ),$
$B\ believes(\ B\ received(H(F(K_{a,b}, Y_a), N_a, F(K_{a,b}, Y_b), N_b))\ \wedge\ B\ believes(N_b, Y_b)\ \wedge$
$A \overset{Ka,b}{\longleftrightarrow} B\ \supset\ A \overset{Ks}{\longleftrightarrow} B\ )$

De las premisas uno a la seis constituyen la suposiciones de inicio de todo protocolo de autenticación. La premisa siete tiene el objetivo de enfatizar el hecho de que si una entidad (A o B) genera algunos valores aleatorios, tal entidad cree en dichos valores. Estas suposiciones se llevan a cabo debido a que el protocolo hace uso del anonimato durante casi toda la ejecución del mismo, y no se pueden establecer relaciones de confianza entre ambas

entidades hasta que cada una recibe el último mensaje. Por lo que cada entidad confiando en sus propios valores generados, calcula otras cantidades en base a ellos (y a otras cantidades que recibe), dando así una cierta garantía de seguridad en dichos cálculos.

De las premisas ocho a la doce se representan la recepción y comprensión de los mensajes en el protocolo. Las premisas trece y catorce constituyen las premisas de interpretación que cada entidad realiza de acuerdo a la especificación del protocolo. En la premisa trece se expresa el hecho de que si una entidad (A o B) cuenta con los elementos necesarios para formar la llave de sesión, entonces se encuentra en posesión de ella. Dentro de este protocolo, que una entidad cuente con los elementos necesarios para generar la llave de sesión no implica que confíe en esta, no es hasta que recibe (y verifica) el mensaje de confirmación cuando una entidad podrá confiar en la llave de sesión. Este hecho se enfatiza en la premisa catorce.

Se puede observar (desde el protocolo y las premisas) que la autenticación por EAP-PWD presenta una simetría en sus intercambios de mensajes y operaciones ya que ambas entidades realizan las mismas funciones de acuerdo a los parámetros que reciben. Con base en esto, sólo se presentarán las derivaciones para la entidad A. Las derivaciones para la entidad B pueden obtenerse siguiendo los mismos pasos (y razonamientos) presentados para A.

Derivaciones para el participante A

A1 A *believes* A $has(\langle N_b \rangle_{*A}, \langle F(K_{a,b}, Y_b) \rangle_{*A})$
Por P9, Ax10, N, Ax1, MP

A2 A *believes* A $has(F(K_{a,b}, Y_a))$
Por P2, P5, Ax12, N, Ax1, MP

A3 A *believes* A $has(K_s)$
Por A1, A2, P2, P3, P13, Ax1, MP

A4 A *believes* A $received(H(F(K_{a,b}, Y_b), N_b, F(K_{a,b}, Y_a), N_a))$
Por P11, A3, Ax8, N, Ax1, MP

A5 A *believes* $A \overset{Ks}{\longleftrightarrow} B$
Por A4, P7, Ax2, P1, P14, Ax1, MP

A6 A *believes* $A \overset{Ks-}{\longleftrightarrow} B$ **G3**
Por A3, A5

A7 A *believes* B $said(\ H(F(K_{a,b}, Y_b), N_b, F(K_{a,b}, Y_a), N_a)\ )$
Por A5, P11, Ax3, N, Ax1, MP

A8 A *believes* B $says(\ H(F(K_{a,b}, Y_b), N_b, F(K_{a,b}, Y_a), N_a)\ )$ **G1, G2**
Por P4, P6, Ax18, A7, Ax19, N, Ax1, MP

A9 A *believes*$(\ A \overset{Ks-}{\longleftrightarrow} B \ \wedge\ B$ *says* $\{H(F(K_{a,b}, Y_b), N_b, F(K_{a,b}, Y_a), N_a)\}_{Ks}\ )$ **G4**
Por A6, A8

A10 *A believes* $fresh(K_s)$ **G5**
P4, P6, Ax18, N, Ax1, MP

De acuerdo a los resultados obtenidos por el análisis para la entidad A, se puede ver que esta entidad cumple con los objetivos esenciales de autenticación. De igual manera, si se realizara las derivaciones para la entidad B, se podría demostrar que este cumple con todas la metas preestablecidas debido a la simetría del protocolo. Con esto queda demostrado que el protocolo es seguro. Al final de la ejecución, cada entidad demuestra estar presente, además de identificarse y establecer una llave de sesión.

Como se ha mencionado (y como se observó en el análisis), las relaciones de confianza para EAP-PWD sólo se dan hasta que cada entidad manda el mensaje de confirmación empleando la llave de sesión y los parámetros intercambiados. Cada entidad verifica la función H en base a la llave de sesión y la información recibida, y es en ese momento que tanto A como B pueden saber si realmente establecieron contacto con la entidad indicada en un inicio. Esto puede constituir una desventaja ya que ambas entidades requieren ejecutar todos los pasos del protocolo para verificar si cada entidad es quien dice ser, dejando abierta la posibilidad para que un adversario lleve a cabo un ataque de denegación de servicios.

En general, con este protocolo se puede ver las ventajas en cuestión de seguridad que presentan los protocolos basados en sistemas de cifrado asimétrico. El hecho de utilizar el problema del logaritmo discreto constituye una medida fuerte de protección, pero presenta la desventaja de tener un alto costo computacional, por lo que sólo se debe emplear en entidades que posean el poder de cómputo necesario para llevar a cabo operaciones de cifrado no convencional.

4.2.2. Acceso y autenticación en sistemas LTE

Para el estándar LTE, el protocolo de autenticación empleado para identificar y verificar a un dispositivo móvil (UE bajo la terminología de LTE) es el denominado EPS-AKA (*Evolved Packet System AKA*). Este protocolo pertenece a la familia de protocolos derivada a partir del protocolo de Autenticación y Acuerdo de Llaves (AKA por sus siglas en inglés) y se trata de un protocolo de acceso a la red (LTE). Antes de describir y analizar a detalle el protocolo EPS-AKA, es necesario explicar algunos aspectos básicos de la arquitectura LTE.

Tal como se ve en la figura 4.1, una red LTE se encuentra constituida por dos redes principales: la red de acceso, conocida en LTE como E-UTRAN (*Evolved Universal Terrestial Radio Access Network*); y el núcleo de la red, al cual se le denomina como EPC (*Evolved Packet Core*) [52]. E-UTRAN esta constituida por una red de estaciones base llamadas en LTE Evolved Node Base stations (eNBs). El núcleo de la red LTE consiste de algunos elementos,

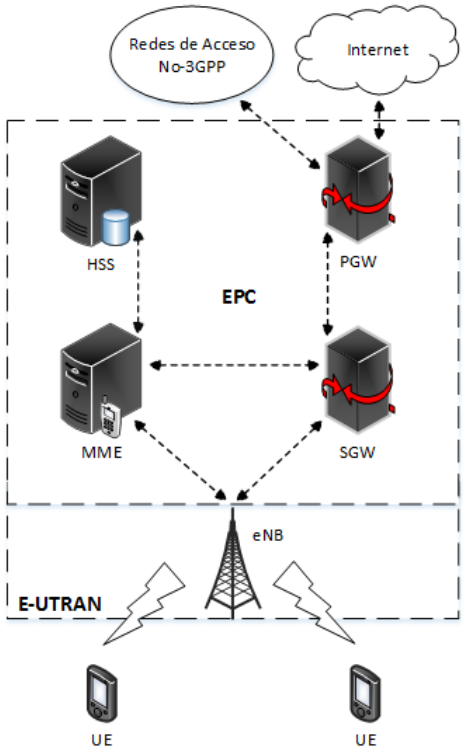

Figura 4.1: Arquitectura LTE

entre estos tenemos al MME (*Mobility Management Entity*) junto a el SGW (*Serving Gateway*), así como el PDN GW (*Packet Data Network Gateway*) y el HSS (*Home Subscriber Server*). Todos estos elementos se comunican de forma segura mediante el establecimiento de túneles cifrados creados por ejemplo a través de IP seguro (IPsec) [11].

El comité 3GPP ha definido toda una arquitectura de seguridad para LTE. Dentro de esta arquitectura se encuentran definidos cinco niveles de seguridad [53] los cuales están enfocados a diversos aspectos como lo son la seguridad en el acceso a la red, la seguridad en el dominio de la red, la seguridad en el dominio del usuario, la seguridad en el dominio de las aplicaciones y la seguridad en el dominio fuera de 3GPP (no 3GPP). El presente trabajo sólo se enfoca en el aspecto de seguridad en el acceso, que es en donde se emplea el protocolo EPS-AKA.

En el ambiente de LTE, un dispositivo móvil (UE) se encuentra constituido por dos entidades autónomas, la primera es el USIM (*Universal Subscriber Identity Module*), conocido en la cultura popular como "Tarjeta SIM", y la segunda se trata del Equipo Móvil (ME por sus siglas en inglés), que no es otra cosa que el "Equipo Celular". Ambas entidades se comunican entre sí con la finalidad de cooperar en los procesos de autenticación. Dentro del módulo USIM, se encuentran almacenados algunos datos requeridos para la autenticación un usuario

que posea una tarjeta SIM, entre los cuales destacan los siguientes:

- IMSI: Es el acrónimo de *International Mobile Subscriber Identity* (Identidad Internacional del Abonado a un Móvil). Consiste de una clave numérica cuya longitud máxima es de quince dígitos que identifica de manera única (a nivel internacional) a un módulo o tarjeta SIM. Este número permite la identificación de la tarjeta SIM a través de las redes de telefonía celular como GSM, UMTS y LTE.

- K: Es una llave criptográfica permanente que sólo es compartida entre el USIM y el HSS. Su longitud es de 128 bits y nunca es transferida fuera de las entidades mencionadas (ni siquiera el propietario del dispositivo móvil que contiene una tarjeta SIM conoce su valor). Esta llave representa un punto clave para llevar a cabo el proceso de autenticación.

- SQN: Es un número de secuencia que se emplea para asegurar que los mensajes enviados son recientes. Su longitud es de 48 bits y su valor debe estar sincronizado entre el USIM y el HSS.

El proceso de autenticación mutua entre el UE y el EPC es la característica más importante dentro del sistema de seguridad en LTE. El sistema LTE utiliza como base el protocolo AKA para lograr la autenticación mutua entre el UE y el EPC y generar además una llave para el cifrado (CK) y una llave de integridad (IK), las cuales son empleadas a su vez para derivar algunas otras llaves de sesión distintas para el cifrado y la protección de la integridad de los mensajes subsecuentes [54].

Cuando un UE se conecta al EPC a través de E-UTRAN, la entidad MME representa al núcleo de la red para llevar a cabo el procedimiento de autenticación con el equipo usuario (UE) mediante el protocolo EPS-AKA. El procedimiento inicia cuando el eNB envía un petición de identidad de usuario al UE (ver figura 4.2), luego se siguen los siguientes pasos:

1. Cuando el UE recibe la petición de identidad de usuario, este envía su identidad (IMSI) como respuesta al eNB el cual lo reenvía al MME.

2. El MME envía el IMSI recibido junto con el identificador de la red de servicio a la que pertenece (SN_{id}) al HSS en un mensaje de petición de autenticación.

3. Cuando el HSS recibe los datos de la petición de autenticación, el HSS utiliza la llave K (compartida sólo con el UE) y dos tipos de funciones especiales (denominadas f_1 y f_2 para autenticación de mensajes y f_3, f_4 y f_5 para generación de llaves) para generar una lista de vectores de autenticación EPS. Estos vectores se construyen como sigue:

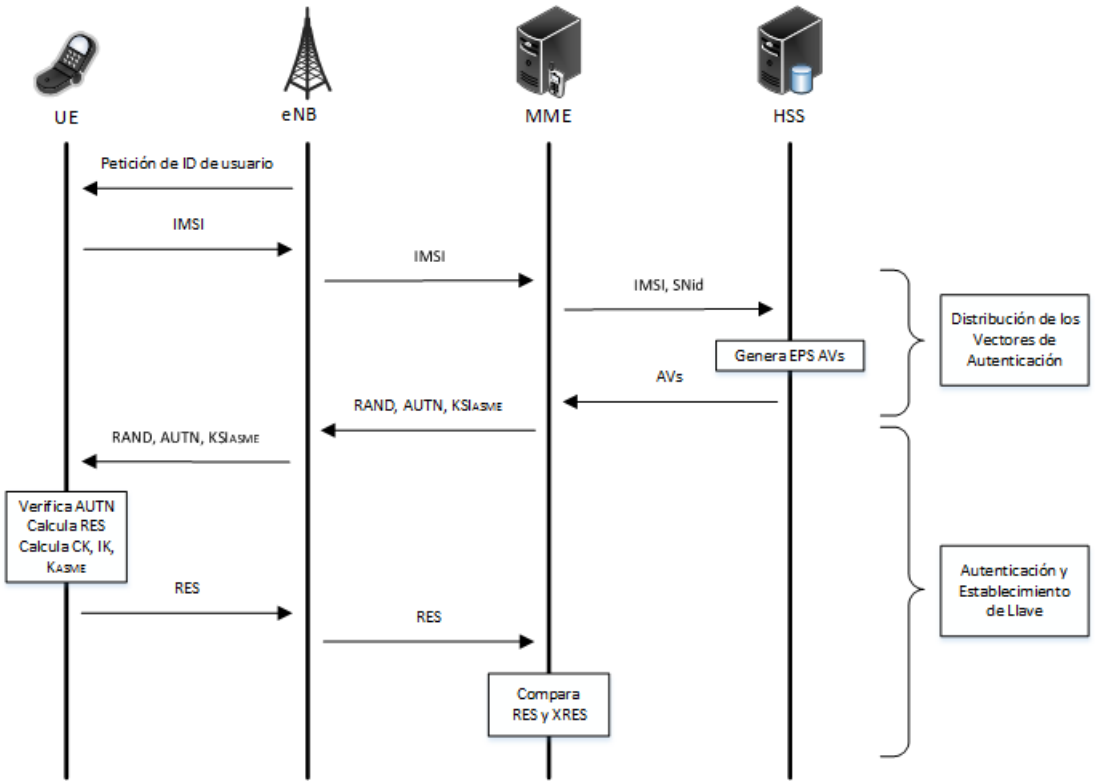

Figura 4.2: Protocolo de autenticación EPS-AKA

- HSS genera un nonce aleatorio denominado $RAND$. Con el, HSS calcula el código de autenticación de mensaje $MAC = f_1(K, SQN_h, RAND)$; la respuesta esperada $XRES = f_2(K, RAND)$; la llave de cifrado $CK = f_3(K, RAND)$; la llave de de integridad $IK = f_4(K, RAND)$; la llave de protección $AK = f_5(K, RAND)$.
- HSS calcula además el token de autenticación $AUTN = (SQN_h \oplus AK, MAC)$; una llave de sesión generada a partir de una función de derivación de llave (KDF por sus siglas en inglés) $K_{ASME} = KDF(SQN_h \oplus AK, CK, IK, SN_{id})$ y finalmente genera el vector de autenticación $AV = (RAND, AUTN, XRES, K_{ASME})$.
- HSS repite el mismo procedimiento incrementando SQN_h y generando un nuevo valor aleatorio $RAND$ por cada vector de autenticación que construya.

4. HSS envía los vectores de autenticación (AVs) a MME como respuesta a la petición de autenticación de datos.

5. Una vez que se reciben los vectores de autenticación de HSS, MME sólo envía $RAND$ y $AUTN$ junto con un identificador de llave KSI_{ASME} a UE en un mensaje de petición de autenticación. El UE y el MME utilizan el identificador KSI_{ASME} para identificar la llave K_{ASME}.

6. El UE recibe el mensaje, extrae SQN_h (utilizando la llave AK obtenida a partir de $RAND$), calcula el valor MAC esperado ($XMAC$) y verifica que su valor coincida con el del MAC recibido en $AUTN$. Si no poseen el mismo valor, ocurre un fallo de integridad y envía un mensaje error. Si coinciden, UE verifica ahora el valor del SQN_h recibido con el valor del SQN_e almacenado en su memoria, invocando a un proceso de sincronización en caso de que los valores no coincidan o no estén dentro de un rango permitido. Si SQN_h y SQN_e son válidos, EU procede a calcular (de la misma manera como lo hizo HSS) CK, IK, RES y K_{ASME}, enviando RES como respuesta a la petición de autenticación.

7. El MME verifica que el valor de RES recibido de UE y el valor $XRES$ recibido de HSS sean iguales, de ser así HSS le envía un mensaje de éxito a UE finalizando el proceso de autenticación mediante EPS-AKA.

Con respecto a las funciones f_1, f_2, f_3, f_4 y f_5, se trata de funciones cuya base es utilizar llaves de 128 bits con algoritmos de cifrado simétrico. Además, como buen requerimiento de seguridad, estas funciones deben estar construidas de tal forma que sea inviable (bajo cualquier poder computacional) calcular la llave K a partir del conocimiento de sus entradas y salida. En el estándar LTE, se deja abierto el uso de diversos algoritmos para implementarlas, pero deben cumplir con los requerimientos recién mencionadas. No obstante, en el documento TS35.205 [55] se presenta un ejemplo de algoritmos para la implementación de estas funciones, estos algoritmos son conocidos como "Algoritmos de Milenage", los cuales se basan en la utilización de AES de 128 bits.

De el lado de el usuario (UE), los cálculos de las funciones f_1, f_2, f_3, f_4 y f_5 son ejecutadas por la USIM, permitiendo así que la llave K nunca salga este módulo. Por otra parte, la llave K_{ASME} es derivada por el ME (fuera del módulo USIM) siguiendo el mismo procedimiento empleado por HSS. Para mayor referencia de la derivación y uso de la llave K_{ASME} se puede consultar [11].

Acerca del mecanismo de verificación y generación del número de secuencia SQN, podemos decir que no se encuentra estandarizado. En el documento TS33.102 [56] se dan algunos ejemplos de como se puede implementar este mecanismo.

Habiendo explicado las bases y el procedimiento de autenticación de usuarios en LTE, ahora es posible obtener una idea abstracta del protocolo para realizar el análisis formal del mismo. Es importante hacer notar que una idea abstracta del protocolo EPS-AKA no ha sido reportada en la literatura de autenticación, por lo que el modelo presentado a continuación es propio de este trabajo.

4.2.2.1. Análisis formal del protocolo EPS-AKA

El primer paso para analizar el protocolo EPS-AKA con la lógica SVO es obtener una idea abstracta del mismo. Para realizar esta labor, es necesario tener una idea clara de las operaciones que se realizan dentro de este, así como entender el objetivo de cada parámetro enviado en el intercambio de mensajes. Como es común en todo protocolo de comunicaciones, parte de los datos enviados en los mensajes tienen carácter informativo, por lo que este tipo de información se puede omitir.

De acuerdo a la descripción expuesta, y modelando a UE, MME y HSS como las entidades A, B y S respectivamente, el protocolo EPS-AKA realiza lo siguiente:

1. $A \rightarrow B: \quad ID_a$
2. $B \rightarrow S: \quad ID_a, ID_b$
3. $S \rightarrow B: \quad N_s, F_2(K_{a,s}, N_s), F_{KDF}(K_{a,s}, SQN, ID_b, N_s), \{SQN\}_{Ka,s}, F_1(K_{a,s}, SQN, N_s)$
4. $B \rightarrow A: \quad N_s, \{SQN\}_{Ka,s}, F_1(K_{a,s}, SQN, N_s)$
5. $A \rightarrow B: \quad F_2(K_{a,s}, N_s)$

En donde $ID_a = IMSI$, $ID_b = SN_{id}$, $N_s = RAND$, $K_{a,s} = K$, $SQN = SQN_h$. Las funciones F_1 y F_2 representan a f_1 y f_2 respectivamente. F_{KDF} es la función de derivación de llave para generar K_{ASME}. Se han omitido la representación de las funciones f_3 y f_4 debido a que ambas dependen de K ($K_{a,s}$) y $RAND$ (N_s) y contribuyen posteriormente a la formación de K_{ASME} ($F_{KDF}(K_{a,s}, SQN, ID_b, N_s)$). De igual manera, se omite la representación de la llave AK (salida de la función f_5) que se utiliza para cifrar el número de secuencia SQN_h (SQN), ya que es finalmente, una función de la llave K ($K_{a,s}$). De ahí que su representación sea $\{SQN\}_{Ka,s}$.

Esta es una buena primera aproximación. Se puede observar en el mensaje tres todos los elementos que constituyen al vector de autenticación (AV). Aquí,

$N_s = RAND$,
$XRES = F_2(K_{a,s}, N_s)$,
$K_{ASME} = F_{KDF}(K_{a,s}, SQN, ID_b, N_s)$ y
$AUTN = (\{SQN\}_{Ka,s}, F_1(K_{a,s}, SQN, N_s))$.

No obstante, existen ciertos aspectos que aún faltan incluir en este modelo. Como se puede apreciar en el mensaje tres, elementos como $F_{KDF}(K_{a,s}, SQN, ID_b, N_s)$ que como tal es la llave K_{ASME}, se envía sin cifrar. Cualquier adversario podría obtener la llave de cifrado interviniendo en el canal entre el MME (B) de la estación de servicio y el HSS (S) en donde se encuentra registrado el usuario. La consideración que hace falta para esta idealización es el hecho de que todas las comunicaciones entre los elementos del núcleo de la red son seguras

y se establecen mediante mecanismos de cifrado asimétrico como IPsec [11]. Con ello, es muy difícil para un adversario poder obtener la llave K_{ASME}. Por lo tanto, una posible (y burda) aproximación sería representar a todo el mensaje tres como un mensaje cifrado con la una llave compartida entre MME y HSS (generada a partir de la ejecución de IPsec). Esto sería:

3. $S \rightarrow B: \quad \{N_s, F_2(K_{a,s}, N_s), K_{asme}, \{SQN\}_{Ka,s}, F_1(K_{a,s}, SQN, N_s)\}_{Kbs}$

Con esta modificación, el modelo cubre de manera más precisa lo que ocurre durante la ejecución del protocolo. Es importante mencionar que el mecanismo para establecer un canal seguro entre MME y HSS lleva consigo un completo (y posiblemente complejo) proceso de autenticación, el cual se encuentra fuera del objetivo de autenticar entidades mediante el protocolo AKA. Considerando este detalle, se puede realizar una modificación a la idealización del protocolo recién expuesta en la que MME y HSS son consideradas como una sola entidad. Esta idea va además acorde con el hecho de que MME y HSS forman parte del núcleo del sistema LTE (EPC) y de que MME guarda un conjunto de vectores de autenticación para autenticar al usuario posteriormente sin la necesidad de comunicarse con HSS. Esta acción hace que la abstracción del protocolo sea aun más sencilla y fácil de analizar, quedando establecida de la siguiente manera:

1. $A \rightarrow B: \quad ID_a$
2. $B \rightarrow A: \quad N_b, \{SQN\}_{Ka,b}, F_1(K_{a,b}, SQN, N_b)$
3. $A \rightarrow B: \quad F_2(K_{a,b}, N_b)$

Si bien esta simplificación oculta algunos de los detalles de los parámetros empleados durante la ejecución del protocolo, es funcionalmente adecuada debido a que se incluyen los datos necesarios para establecer la autenticación de ambas entidades. Esto se verá más claramente con el análisis formal que se presenta a continuación.

Premisas iniciales

P1 $A\ believes (A \overset{Ka,b}{\longleftrightarrow} B)$, $B\ believes (A \overset{Ka,b}{\longleftrightarrow} B)$
P2 $A\ believes\ fresh(SQN)$, $B\ believes\ fresh(SQN)$
P3 $A\ believes\ A\ has(K_{a,b})$, $B\ believes\ B\ has(K_{a,b})$
P4 $B\ believes\ fresh(N_b)$
P5 $B\ received(\ ID_a\ )$
P6 $A\ received(\ N_b, \{SQN\}_{Ka,b}, F_1(K_{a,b}, SQN, N_b)\ )$
P7 $B\ received(\ F_2(K_{a,b}, N_b)\ )$
P8 $B\ believes\ B\ received(\ ID_a\ )$
P9 $A\ believes\ A\ received(\ \langle N_b \rangle_{*A}, \{SQN\}_{Ka,b}, F_1(K_{a,b}, SQN, \langle N_b \rangle_{*A})\ )$

P10 $B\ believes\ B\ received(\ F_2(K_{a,b}, N_b)\ )$
P11 $A\ believes(\ B\ says(SQN, \langle N_b \rangle_{*A})\ \wedge\ A\ has(K_{a,b}) \supset\ B\ says(A \overset{Ks}{\longleftrightarrow} B, fresh(K_s))\ )$
P12 $A\ believes\ B\ controls(A \overset{Ks}{\longleftrightarrow} B)$
P13 $A\ believes\ B\ controls\ fresh(K_s)$
P14 $A\ believes(\ A\ has(SQN, N_b)\ \wedge\ A\ has(K_{a,b})\ \supset\ A\ has(K_s)\ )$
P15 $B\ believes(A \overset{Ks}{\longleftrightarrow} B)$
P16 $B\ believes\ fresh(K_s)$
P17 $B\ believes\ B\ has(K_s)$

Para este análisis, K_s representa la llave de sesión generada para una ejecución específica del protocolo. Las premisas uno y tres representan el hecho de que A y B comparten un secreto previamente establecido. En la premisa dos, el hecho de que A y B conocen que el valor SQN es reciente se debe a que ambas entidades mantienen una sincronía de su valor tal como se menciona en la especificación del protocolo. En la premisa cuatro, se indica que B ha generado un valor aleatorio para la ejecución actual de EPS-AKA. Las premisas cinco, seis y siete corresponden a las premisas de recibimiento mientras que las premisas ocho, nueve y diez corresponden a las premisas de comprensión.

Un punto clave del análisis recae en la premisa once, ya que es en esta donde se introduce la llave de sesión K_s. Como se indica en la especificación del protocolo, esta llave nunca se transfiere ni se utiliza de manera explícita durante la ejecución del mismo. A calcula esta llave de acuerdo a los parámetros SQN y N_b (y $K_{a,b}$), los cuales son generados por B, por lo que si B le envía estos datos a A, y A al recibirlos comprueba que son íntegros y que fueron emitidos por B para la actual ejecución del protocolo, entonces A puede interpretar que B le indica la existencia de una llave de sesión K_s reciente. A partir de esta premisa, A puede inferir que existe una llave de sesión la cual se encuentra bajo la jurisdicción de B y que además es reciente gracias a SQN (premisas doce y trece). La premisa catorce indica que si A se encuentra en posesión de todos los elementos para formar la llave K_s, entonces A posee dicha llave.

Finalmente, en las premisas quince, dieciseis y diecisiete se enfatiza el hecho de que B es el creador de la llave K_s y por lo tanto la posee, conoce que es reciente y cree en ella.

Derivaciones para el participante A

A1 $A\ believes\ A\ received(\ F_1(K_{a,b}, SQN, \langle N_b \rangle_{*A})\ )$
Por P9, Ax7, N, Ax1, MP
A2 $A\ believes\ B\ said(\ SQN, \langle N_b \rangle_{*A}\ )$
Por A1, P1, Ax3, N, Ax1, MP

A3 *A believes B says*($SQN, \langle N_b \rangle_{*A}$) **G1**
Por A2, P2, Ax17, Ax19, N, Ax1, MP

A4 *A believes B says*($A \overset{Ks}{\longleftrightarrow} B, fresh(K_s)$)
Por A3, P3, P11, Ax1, MP

A5 *A believes* ($A \overset{Ks}{\longleftrightarrow} B$)
Por A4, Ax15, P12, Ax16, N, Ax1, MP

A6 *A believes* $fresh(K_s)$ **G5**
Por A4, Ax15, P13, Ax16, N, Ax1, MP

A7 *A believes A has*(K_s)
Por A1, Ax10, Ax13, N, P3, P14, Ax1, MP

A8 *A believes* ($A \overset{Ks-}{\longleftrightarrow} B$) **G3**
Por A5, A7

Derivaciones para el participante B

B1 *B believes A said*(N_b)
Por P10, P1, Ax3, N, Ax1, MP

B2 *B believes A says*(N_b) **G1,G2**
Por B1, P4, Ax19, N, Ax1, MP

B3 *B believes*($A \overset{Ks-}{\longleftrightarrow} B$) **G3**
Por P15, P17

B4 *B believes* $fresh(K_s)$ **G5**
Por P10

De acuerdo a los resultados obtenidos por el análisis, este protocolo cumple con algunas de las metas necesarias de todo protocolo de autenticación. A y B confirman que ambos se encuentran funcionando durante la ejecución del protocolo (metas **G1** y **G2**). La entidad A no puede cumplir con la meta **G2** debido a que no genera un reto (*nonce*) dentro del protocolo, pero puede demostrar que B se encuentra en ese momento gracias al número de secuencia que recibe. Como ya se mencionó, A y B llevan un registro síncrono del número de secuencia, que se incrementa o cambia de valor por cada ejecución del protocolo, por lo que A puede saber si un mensaje es reciente o no gracias al mismo sin la necesidad de generar un reto para B.

Se puede ver además que de acuerdo al análisis formal, al final de la ejecución del protocolo, ambas entidades se encuentran en posesión de una llave de sesión la cual es reciente (metas **G3** y **G5**). Este es un objetivo para lo que fue diseñado y gracias al análisis SVO se puede demostrar formalmente que se cumple. Un detalle interesante es el hecho de que

ninguna de las dos entidades envía un mensaje para confirmar el conocimiento de la llave de sesión K_s. Es por ello que no es posible derivar la meta **G4**.

Acorde a la especificación, el protocolo EPS-AKA no fue diseñado para establecer una confirmación de la llave de sesión. En su especificación se asume (y se sabe) que el módulo USIM y la entidad HSS son entidades muy confiables las cuales se encuentran protegidas con un alto grado de seguridad. Si bien el módulo USIM es un dispositivo que se encuentra más al alcance de una persona (basta con retirar la tarjeta SIM del equipo móvil), esta implementado de tal manera que los datos almacenados en él no pueden ser accedidos o modificados bajo ningún medio. Con este hecho, es posible confiar en que las entidades UE (A) y HSS (B) calculan la llave de sesión (durante la ejecución de EPS-AKA) sin la necesidad de una confirmación de la misma. Por lo que se puede afirmar que la falta de confirmación de la llave de sesión no representa una vulnerabilidad al protocolo EPS-AKA.

En términos generales, el protocolo EPS-AKA cumple de manera efectiva su cometido. Sin embargo, existen algunos ataques que se pueden llevar a cabo dentro de EPS-AKA que no pueden ser reflejados por los lenguajes formales. Entre estos ataques se puede mencionar la revelación de la identidad del usuario (IMSI) en el mensaje uno del protocolo así como los ataques de denegación de servicios mediante la solicitud de muchos vectores de autenticación al HSS [54]. Estos ataques deben ser tratados mediante herramientas que se encuentran fuera del análisis formal de protocolos. El análisis SVO es sólo una herramienta de apoyo para saber si cumplen o no los objetivos para los que fue diseñado un protocolo de autenticación.

Referencias

[1] W. Stallings, *Crytography and Network Security: Principles and Practice*. Pearson Education, 6ta. ed., 2014.

[2] J. Kurose y K. Ross, *Computer Networking: A Top-Down Approach*. Pearson Education, 6ta. ed., 2013.

[3] J. Katz y Y. Lindell, *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2da. ed., 2014.

[4] "Data Encryption Standart (DES)", FIPS PUB 46-3, Octubre 1999.

[5] J. Daemen y V. Rijmen, "*Rijndael: The Advanced Encryption Standard*", Dr. Dobb's Journal, Marzo 2001.

[6] "The MD5 Message-Digest Algorithm". IETF RFC 1321, Marzo 2013.

[7] "Secure Hash Standard (SHS)". FIPS PUB 180-4, Marzo 2012.

[8] D. H. Krawczyk, M. Bellare, y R. Canetti, "HMAC: Keyed-Hashing for Message Authentication". IETF RFC 2104, Marzo 2013.

[9] "Digital Signature Standard (DSS)". FIPS PUB 186-4, Julio 2013.

[10] R. W. Shirey, "Internet Security Glossary, Version 2". ISE RFC 4949, Marzo 2013.

[11] D. Forsberg, G. Horn, W.-D. Moeller, y V. Niemi, *LTE Security*. Wiley, 2da. ed., 2012.

[12] M. Nakhjiri y Ma. Nakhjiri, *AAA and Network Security for Mobile Access: Radius, Diameter, EAP, PKI, and IP Mobility*. Wiley, 1ra. ed., 2005.

[13] L. Gong, "Variations on the themes of message freshness and replay - or the diculty in devising formal methods to analyze cryptographic protocols", in *Proceedings of the Computer Security Foundations Workshop VI*, pp. 131-136, IEEE Computer Society Press, 1993.

[14] R. M. Needham y M. D. Schroeder, "Using encryption for authentication in large networks of computers", *Commun. ACM*, vol. 21, pp. 993-999, Diciembre 1978.

[15] D. E. Denning y G. M. Sacco, "Timestamps in key distribution protocols", *Commun. ACM*, vol. 24, pp. 533-536, Agosto 1981.

[16] B. C. Neuman y S. G. Stubblebine, "A note on the use of timestamps as nonces", *Operating Systems Review*, vol. 27, no. 2, pp. 10-14, 1993.

[17] W. Diffie y M. E. Hellman, "Multiuser cryptographic techniques", in *Proceedings of the June 7-10, 1976, National Computer Conference and Exposition*, AFIPS '76, (New York, NY, USA), pp. 109-112, ACM, 1976.

[18] T. Y. C. Woo y S. S. Lam, "Authentication for distributed systems", *Computer*, vol. 25, pp. 39-52, Enero 1992.

[19] X. Liu y A. Fapojuwo, "Formal evaluation of major authentication methods for ieee 802.11i wlan standard", in *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, pp. 1-5, Septiembre 2006.

[20] M. Burrows, M. Abadi, y R. Needham, "A logic of authentication", Tech. Rep. 39, Digital Systems Research Center, Febrero 1989.

[21] M. Abadi y M. R. Tuttle, "A semantics for a logic of authentication (extended abstract)", in *Proceedings of the Tenth Annual ACM Symposium on Principles of Distri- buted Computing*, PODC '91, (New York, NY, USA), pp. 201-216, ACM, 1991.

[22] L. Gong, R. Needham, y R. Yahalom, "Reasoning about belief in cryptographic protocols", in *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pp. 234-248, Mayo 1990.

[23] P. van Oorschot, "Extending cryptographic logics of belief to key agreement protocols", in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, CCS '93, (New York, NY, USA), pp. 232-243, ACM, 1993.

[24] P. Syverson y P. Van Oorschot, "On unifying some cryptographic protocol logics", in *Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on*, pp. 14-28, Mayo 1994.

[25] M. Garrido, *Lógica Simbólica*. Tecnos, 2da. ed., 1991.

[26] B. F. Chellas, *Modal Logic: An Introduction*. Cambridge University Press, 1ra. ed., 1980.

[27] A. Almuhaideb, B. Srinivasan, P. D. Le, C. Wilson, y V. Malhotra, "Analysis of mobile authentication protocols by svo logic", in *Proceedings of the First International Conference on Security of Internet of Things*, SecurIT '12, (New York, NY, USA), pp. 126-134, ACM, 2012.

[28] P. Syverson y P. Van Oorschot, "A Unified Cryptographic Protocol Logic", NRL Publication 5540-227, Naval Research Laboratory, 1996.

[29] P. F. Syverson y I. Cervesato, "The logic of authentication protocols", in *Revised Versions of Lectures Given During the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, (London, UK, UK), pp. 63-136, Springer-Verlag, 2001.

[30] M. Abadi y R. Needham, "Prudent engineering practice for cryptographic protocols", *Software Engineering, IEEE Transactions on*, vol. 22, pp. 6-15, Enero 1996.

[31] D. C. Neuman, S. Hartman, K. Raeburn, y T. Yu, "The Kerberos Network Authentication Service (V5)". IETF RFC 4120, Junio 2015.

[32] CCITT. CCITT Blue Book, "Recommendation X.509 and ISO 9594-8: The Directory-Authentication Framework". Geneva, Marzo 1988.

[33] D. Otway y O. Rees, "Efficient and timely mutual authentication", *SIGOPS Oper. Syst. Rev.*, vol. 21, pp. 8-10, Enero 1987.

[34] R. J. Anderson y R. M. Needham, "Robustness principles for public key protocols", in *Proceedings of the 15th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '95, (London, UK, UK), pp. 236-247, Springer-Verlag, 1995.

[35] Massachusetts Institute of Technology,"MIT - Massachusetts Institute of Technology", [En línea]. Disponible en: *http://web.mit.edu/*. [Último acceso: Noviembre 2015].

[36] Internet Engineering Task Force, "Internet Engineering Task Force (IETF)", [En línea]. Disponible en: *https://www.ietf.org/*. [Último acceso: Octubre 2015] .

[37] J. T. Kohl y B. C. Neuman, "The evolution of the kerberos authentication service", pp. 78-94, IEEE Computer Society Press, 1994.

[38] Massachusetts Institute of Technology,"Kerberos: The Network Authentication Protocol", [En línea]. Disponible en: *http://web.mit.edu/kerberos/*. [Último acceso: Noviembre 2015].

[39] J. Migeon, "The MIT Kerberos Administrator's How-to Guide. Protocol, Installation and Single Sign On", MIT Kerberos Consortium, Julio 2008.

[40] J. Linn, "Generic Security Service Application Program Interface Version 2, Update 1". IETF RFC 2743, Marzo 2013.

[41] "The FreeRADIUS Technical Guide", The FreeRADIUS Server Project and Contributors, 2014.

[42] J.-C. Chen y Y.-P. Wang, "Extensible authentication protocol (eap) and ieee 802.1x: tutorial and empirical experience", *Communications Magazine, IEEE*, vol. 43, pp. supl.26-supl.32, Diciembre 2005.

[43] M. Catur Bhakti, A. Abdullah, y L. Jung, "Eap-based authentication with eap method selection mechanism: Simulation design", in *Research and Development, 2007. SCOReD 2007. 5th Student Conference on*, pp. 1-4, Diciembre 2007.

[44] J. Vollbrecht, J. D. Carlson, L. Blunk, D. B. D. A. Ph.D., y H. Levkowetz, "Extensible Authentication Protocol (EAP)". IETF RFC 3748, Marzo 2013.

[45] A. Rubens, C. Rigney, S. Willens, y W. A. Simpson, "Remote Authentication Dial In User Service (RADIUS)". IETF RFC 2865, Marzo 2013.

[46] C. Rigney, "RADIUS Accounting". IETF RFC 2866, Marzo 2013.

[47] P. R. Calhoun y D. B. D. A. Ph.D., "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)". ISE RFC 3579, Octubre 2015.

[48] G. W. Zorn y D. Harkins, "Extensible Authentication Protocol (EAP) Authentication Using Only a Password". IETF RFC 5931, Octubre 2015.

[49] M. Lepinski y D. S. T. Kent, "Additional Die-Hellman Groups for Use with IETF Standards". IETF RFC 5114, Octubre 2015.

[50] L. Chen, "Improved otway rees protocol and its formal verication", in *Communications, Circuits and Systems, 2008. ICCCAS 2008. International Conference on*, pp. 498-501, Mayo 2008.

[51] T. Hansen y D. E. E. 3rd, "US Secure Hash Algorithms (SHA and HMAC-SHA)". IETF RFC 4634, Octubre 2015.

[52] K. Alezabi, F. Hashim, S. Hashim, y B. Ali, "An efficient authentication and key agreement protocol for 4g (lte) networks", in *Region 10 Symposium, 2014 IEEE*, pp. 502-507, Abril 2014.

[53] "3GPP System Architecture Evolution (SAE)"; Security Architecture (Rel 12), 3GPP TS 33.401 V12.14.0, Abril 2015.

[54] J. Cao, M. Ma, H. Li, Y. Zhang, y Z. Luo, "A survey on security aspects for lte and lte-a networks", *Communications Surveys Tutorials, IEEE*, vol. 16, pp. 283-302, 2014.

[55] "Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5* (Rel 12)", 3GPP TS 35.205 V12.0.0, Octubre 2014.

[56] "3G Security", Security Architecture (Rel 12), 3GPP TS 33.102 V12.2.0, Enero 2015.

Conclusiones y trabajo a futuro

Al inicio de esta tesis se realizó una descripción general sobre el problema de autenticación en sistemas de cómputo distribuido y se mostraron algunos de los primeros trabajos en el área. Estos trabajos constituyen el fundamento de los protocolos de autenticación actuales, por lo que su estudio a detalle ayudó a una mejor comprensión de los mismos. Sin embargo, los análisis establecidos en un inicio presentan un carácter no sistemático e informal, en los cuales no se sabe con exactitud que es lo que se busca encontrar en cada protocolo de autenticación.

El hecho de que no se contara con herramientas sistemáticas para el análisis de protocolos, motivó a un conjunto de investigadores a desarrollar herramientas para el análisis formal de protocolos criptográficos, derivándose así las lógicas basadas en el uso de lenguajes formales como la lógica SVO. Dichas herramientas permiten decir si un protocolo cumple con los requerimientos para los que fue diseñado así como encontrar vulnerabilidades. Así mismo, y con la finalidad de evitar ciertos errores comunes en el diseño de protocolos criptográficos, Abadi y Needham propusieron unos principios de diseño muy útiles que sirven como complemento a las herramientas formales de estudio de protocolos.

El trabajo principal de esta tesis fue analizar la seguridad y el diseño de los protocolos de autenticación mediante el uso de las herramientas mencionadas. Si bien estas herramientas no proporcionan una solución completa a todo lo que implica autenticación, proporcionan una buena guía para el estudio de los protocolos de autenticación. Entre los protocolos que fueron analizados se encuentran los siguientes:

1. El Protocolo de autenticación de Needham y Schroeder.

2. El protocolo de autenticación *Station-to-Station* (STS).

3. El protocolo de Acuerdo de Llave Cifrado (EKE por sus siglas en inglés).

4. El Protocolo de autenticación de Otway y Rees.

5. El Protocolo EAP-PWD.

6. El Protocolo EPS-AKA.

En el protocolo de Needham y Schroeder se pudo observar que las vulnerabilidades expuestas de manera informal en el capítulo uno pudieron verificarse con las herramientas formales de análisis. Los resultados de este análisis muestran la carencia que tiene el protocolo para que la entidad que recibe una petición de servicio (entidad B) compruebe que dicha petición es reciente y no se trate de una repetición. Este hecho deja abierto al protocolo a un posible ataque de suplantación de identidad en el caso de que el adversario logre comprometer una llave de sesión que se haya utilizado con anterioridad por alguna entidad honesta. Un par de años posteriores, se propusieron posibles soluciones a este problema mediante el uso de estampas de tiempo, donde finalmente predominó la propuesta del MIT (Kerberos) la cual tiene vigencia en la actualidad. Gracias a la validación temporal que se establece con la ayuda de un sistema síncrono de tiempos, la entidad que provee el servicio es capaz de reconocer si una petición se encuentra fuera del rango aceptable para ser considerada válida.

El protocolo STS constituye un ejemplo claro de la diferencia de seguridad que se puede proporcionar con un esquema de llave pública a un esquema de llave simétrica. Del análisis establecido, se puede ver como este protocolo cumple de manera satisfactoria las metas de autenticación. Gracias a la ayuda de un sistema de certificados, se corrige la deficiencia presentada en el intercambio de llaves Diffie-Hellman y así es posible garantizar con mayor facilidad que las dos entidades involucradas participan en la ejecución del protocolo además de verificar su identidad y establecer una llave de sesión. La desventaja que se presenta en este protocolo, como en todo sistema que utilice cifrado asimétrico, es el procesamiento. Como se sabe, los esquemas de llave asimétrica suelen ser muy costosos en cuestiones de procesamiento computacional, por lo que las entidades involucradas deberán contar con una gran cantidad de recursos para ejecutar las operaciones dentro de este tipo de esquemas. De lo contrario, las entidades participantes quedarán más expuestas a recibir un par de peticiones por parte de un adversario con la finalidad de ocasionar que consuman rápidamente sus recursos y generar así una denegación de servicios.

Cuando se expusieron los principios de diseño de protocolos de Abadi y Needham, todo parecía indicar que estos principios serían reglas que no podían pasarse por alto en todo diseño de protocolo de autenticación. Sin embargo, y de acuerdo a las características vistas en el protocolo EKE, se puede asegurar que no en todos los casos es realmente necesario seguir con apego la característica principal de los principios de diseño, la cual dicta que cada mensaje debe ser auto-contenido y lo suficientemente explícito con el fin de no caer en malas interpretaciones de los mismos. Bajo algunas circunstancias, mantener el anonimato de los mensajes puede resultar benéfico, y es ahí donde el paradigma de los principios de diseño se rompe, por lo que es posible omitir el uso de estos siempre y cuando se tomen las

precauciones necesarias.

Con la ayuda del protocolo de Otway y Rees, se pudo demostrar la utilidad de las herramientas de análisis y diseño expuestas en el capítulo dos. En primer lugar, se estudió la propuesta original de Otway y Rees, realizándose su análisis correspondiente con la lógica SVO. De los resultados obtenidos se demostró que el protocolo no proporciona ninguna forma de autenticación, y que sólo se limita a la distribución de la llave de sesión. Una propuesta de mejora a este protocolo fue presentada por Abadi y Needham para ejemplificar el uso de algunos de sus principios. Como se pudo observar, la propuesta de mejora de Abadi y Needham si bien mejora el protocolo en cuestión de procesamiento computacional, lo deja expuesto a un ataque de ejecuciones múltiples del protocolo (*interleaving* en inglés) en el que la entidad que solicita el servicio (entidad A), puede terminar creyendo que realmente ejecutó el proceso con la entidad deseada (entidad B).

En años más recientes, Li Chen publicó en el 2008 una mejora al protocolo de Otway y Rees basándose en la propuesta original de dicho protocolo. Del análisis formal establecido, se observó que la mejora realizada por Li Chen ayuda a que el protocolo cumpla con todas las metas requeridas para que sea considerado como un protocolo de autenticación seguro. Empleando los principios de Abadi y Needham, se procedió a hacer algunas pequeñas correcciones a la propuesta de Li Chen en las que básicamente se eliminaron las operaciones redundantes que ésta presenta. Las correcciones propuestas no afectan de ninguna forma la funcionalidad del protocolo, por lo que el análisis lógico conduce a los mismos resultados.

Un detalle interesante que comúnmente se pasa por alto a la hora de diseñar un protocolo de autenticación es el de validar si dicho protocolo puede contrarrestar o evitar en cierta medida los ataques de denegación de servicios. A pesar de que la propuesta de Li Chen (e inclusive la versión mejorada del mismo propuesta en esta tesis) lleva a cabo un buen proceso de autenticación, depende de que se ejecuten los cinco pasos para verificar si la entidad que solicita el servicio (entidad A) es legítima. Esto puede generar que un adversario que repita el mensaje uno haciéndose pasar por A, provoque que tanto el centro de distribución de llaves (S) como la entidad proveedora del servicio (B) gasten tiempo y recursos atendiendo la petición repetida. Para tratar de hacerle frente a este problema, se propusieron dos versiones ligeramente distintas al protocolo mejorado a partir de la propuesta de Li Chen. En estas versiones, se propone el uso de estampas de tiempo o de contadores con la finalidad de que la entidad S pueda detectar que el mensaje está caducado o de que se trata de una repetición, pudiendo terminar así el proceso sin la necesidad de realizar los pasos tres, cuatro y cinco del protocolo, en los que se realizan operaciones de cifrado. La idea de emplear estampas de tiempo o contadores para ayudar al protocolo a detectar repeticiones esta basada en algunas de las recomendaciones establecidas en los principios de diseño de Abadi y Needham. Para

la lógica SVO resulta transparente el hecho de usar un valor aleatorio, estampa de tiempo o contador como referencia de frescura, quedándose en cierta forma limitada para poder detectar las ventajas y desventajas propiciadas por cada tipo de referencia temporal. Por esta razón, se procedió a realizar una evaluación de carácter argumentativo en la que se mencionan los pros y los contras de las estampas de tiempo o contadores como referencias para hacerle frente a los ataques de negación de servicios, concluyendo que el uso de contadores resulta ser una opción más sencilla y eficaz dentro de el esquema de Otway y Rees.

En cuanto a los protocolos de ambientes inalámbricos, se realizaron los análisis correspondientes para los protocolos EAP-PWD (dentro de Wi-Fi) y EPS-AKA (en LTE). El primer paso que se hizo para llevar a cabo los respectivos análisis, fue plantear y proponer un modelo abstracto de ambos protocolos. En el caso de EAP-PWD, el análisis formal con SVO indica que el protocolo cumple con las metas principales que son la autenticación, establecimiento de llave de sesión y confirmación de la misma, por lo que puede decirse que se trata de un protocolo seguro. Entre los detalles que se pueden observar para este protocolo, es que la toda las comprobaciones de las metas se logran hasta que ambas entidades reciben el último mensaje, ya que este protocolo utiliza el anonimato de los mensajes como base para efectuar el proceso de autenticación (tal como ocurría en el protocolo EKE). Para el caso del protocolo EPS-AKA, los resultados obtenidos por el análisis SVO señalan que el protocolo cumple de manera segura con la autenticación y el establecimiento de llave, pero no se lleva a cabo una confirmación de llave de sesión establecida. El hecho de que no se confirme la llave de sesión, no significa que el protocolo será vulnerable, ya que dicha confirmación se puede establecer en intercambios de mensajes posteriores al protocolo (que es como ocurre realmente dentro de los sistemas LTE).

Con la finalidad de obtener una idea clara y concisa de como funcionan los protocolos de autenticación en ambientes de ejecución real, se montaron dos escenarios de pruebas utilizando equipos físicos de cómputo (configurados bajo el entorno Linux). En el primer escenario, se probó la funcionalidad del protocolo Kerberos versión 5. Para esto, se configuró un servidor Kerberos (Centro de Distribución de Llaves) y se elaboraron una par de aplicaciones cliente/servidor para un servicio de "Eco" autenticado. Se observó la complejidad y robustez del protocolo Kerberos con la ayuda de la herramienta de análisis de paquetes Wireshark. Para el segundo escenario, se empleó un dispositivo móvil y un punto de acceso inalámbrico con el firmware necesario para efectuar autenticación mediante RADIUS-EAP y se procedió a desarrollar el servidor RADIUS (incluyendo el servidor EAP y el módulo de autenticación EAP-PWD) siguiendo las especificaciones de la IETF. El objetivo de este experimento fue observar el conjunto de pasos necesarios para otorgarle acceso a la red a un dispositivo móvil que se conecta por Wi-Fi (IEEE 802.11). Los resultados para esta prueba fueron positivos,

comprobándose además como los fabricantes de dispositivos con acceso a la red no cubren en su totalidad los requerimientos establecidos por los estándares.

Finalmente, como trabajo a futuro se proponen las siguientes áreas:

1. Lenguajes formales. A lo largo de este trabajo se fueron presentando protocolos de autenticación con sus respectivos análisis formales establecidos a través de la lógica SVO. De los análisis y reflexiones realizadas para algunos de los protocolos, se pudo observar como la herramienta de análisis SVO presenta ciertas limitaciones en lo que concierne a seguridad. Se requiere más expresividad por parte de la misma de tal manera que se puedan establecer mejores relaciones de causalidad y de temporización con la finalidad de poder establecer si un protocolo puede resistir a los ataques de ejecución paralela (*interleavings* en inglés).

2. Estrategias de cifrado ligeras. Si bien en esta tesis la mayoría de los protocolos se explicaron a un nivel abstracto en donde se asumía que los sistemas criptográficos empleados eran seguros, se enfatizó el impacto que tiene el utilizar sistemas criptográficos que demanden una gran cantidad de recursos a las entidades involucradas. Teniendo protocolos de autenticación que utilicen algoritmos ligeros de cifrado es posible hacerle frente de mejor manera a los ataques en los que el adversario envía muchas peticiones a las entidades honestas para provocar que consuman sus recursos y generar así denegación de servicios.

3. Nuevos paradigmas de autenticación. En el capítulo uno se presentaron las siete formas básicas de autenticación, dentro de las cuales caen cada uno de los protocolos presentados en este trabajo. Todas estas formas de autenticación se encuentran establecidas para las arquitecturas de tipo cliente/servidor. En la actualidad, con el auge de las arquitecturas *peer-to-peer* (par a par), se requieren de nuevas formas de autenticación en las que se considere la carga generada por la distribución de la estructura de la red.