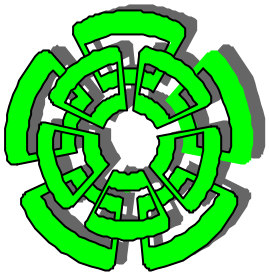

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA
SECCIÓN DE COMPUTACIÓN

Sistema de Seguridad para Intercambio de Datos en Dispositivos Móviles

Tesis que presenta el:

Ing. Carlos Eduardo López Peza

para obtener el grado de Maestro en Ciencias en la especialidad de Ingeniería Eléctrica Opción Computación

Director de la tesis

Dr. Francisco José Rambó Rodríguez Henríquez

México, D.F. ***Abril de 2005***

Índice general

Agradecimientos

Agradezco al Consejo Nacional de Ciencia y Tecnología por el respaldo financiero otorgado durante el programa de maestría. Así como el apoyo brindado para la conclusión de este trabajo de Tesis por medio del proyecto CONACyT 45306.

Agradezco a los profesores de la sección de computación del CINVESTAV-IPN por brindarme sus conocimientos.

A mi asesor por toda la paciencia que me tuvo durante el desarrollo de la tesis, por todas las correcciones en mi investigación y por la enseñanza que me dejo realizar este trabajo de tesis.

A Dios y a la Virgen de Guadalupe por ser mi guía durante toda mi vida.

Agradezco a mis padres, en especial a mi madre Martha Peza Rocha por el apoyo, la comprensión, los ánimos, el buen ejemplo que siempre me ha dado y sobre todo por todo su amor, que sobra decirlo, pero que es infinitamente correspondido. También a mi hermana Yadira y a su esposo Alfredo por darme ánimos y apoyarme siempre y a mis dos sobrinos Adrián y Edgar por ser tan latosos y graciosos.

A Adriana por su comprensión, su amor, sus consejos, por ser mi inspiración y en muchas ocasiones mi guía en los momentos más difíciles de la maestría.831266 gracias!

Agradezco a Lorena por su amistad, por sus consejos y su invaluable ayuda durante toda la maestría.

A los revisores de mi trabajo, el Dr. Luis Gerardo de la Fraga y el Dr. Miguel Ángel León Chávez, por sus comentarios ya que estos contribuyeron a mejorar el contenido de la tesis.

A todo el personal administrativo de la sección de computación, en especial a Sofía Reza por preocuparse tanto por nosotros y por ser en muchas ocasiones como una mamá para cada uno.

A todos mis compañeros de generación porque compartieron conmigo su amistad y su solidaridad en los momentos más difíciles.

A Joselito por toda su ayuda.

De manera muy especial agradezco a mis abuelos por todas sus enseñanzas y su cariño, y sigo agradeciéndole a mi abuelo por seguir velando por toda su familia en donde quiera que este.

Resumen

Diariamente más dispositivos móviles (como teléfonos celulares, PDAs, tarjetas inteligentes, por mencionar algunos) se conectan a redes inalámbricas ya sea para consultar información o realizar transacciones comerciales. Sin embargo, los canales de comunicación inalámbrica son aún más inseguros que los tradicionales -ya que al ser el aire el medio por el cual viaja la información cualquier persona está en posibilidad de alterarla y/o robarla-, por lo que es necesario proveer herramientas altamente confiables y eficientes que permitan proteger la información y así satisfacer las demandas de seguridad de los usuarios.

En esta tesis se presenta un sistema para el intercambio seguro de datos en ambientes inalámbricos enfocado a dispositivos móviles. El sistema está basado en el paradigma cliente - servidor, donde el cliente es un dispositivo móvil. Se utilizó un protocolo inspirado en el Wireless Transport Layer Security (WTLS) y Transport Layer Security (TLS), para establecer sesiones seguras con base a un protocolo de negociación con el cual se establecen los parámetros criptográficos que se emplearán a lo largo de la sesión de intercambio de información. Con estos parámetros se genera una llave de sesión, utilizando los criptosistemas de llave pública soportados por WTLS: Criptosistema de Curvas Elípticas (ECC, por sus siglas en inglés) y RSA. En la fase de intercambio de información el sistema utiliza dicha llave de sesión para cifrar/descifrar los datos entre el cliente y el servidor utilizando algoritmos conocidos de cifrado simétrico tales como: DES, AES, TDES, para proveer confidencialidad. Asimismo, el sistema ofrece los siguientes servicios de seguridad: integridad y autenticación utilizando firma/verificación digital y certificados digitales.

Nuestros resultados confirman que ECC es la opción criptográfica de llave pública más eficiente para implementaciones en dispositivos móviles con restricciones computacionales. Además, mostramos que ECC otorga niveles de seguridad similares a los de RSA, pero con llaves hasta diez veces menores que las utilizadas por RSA. Las pruebas hechas con el sistema de intercambio de información muestran un menor costo computacional dentro del dispositivo móvil debido a la utilización de ECC y un esquema híbrido (ECC y

RSA) en la etapa de negociación y a la utilización de criptografía simétrica en la etapa de intercambio.

De manera concreta, las principales aportaciones de este trabajo son:

Implementar el protocolo de negociación de WTLS para ambientes inalámbricos.

Implementar un modelo híbrido combinando las ventajas de ECC y RSA para eficientar el proceso de autenticación.

Desarrollar el modelo analítico del protocolo de negociación, esto permitió estimar el tiempo de duración de este protocolo.

Abstract

Nowadays, more and more mobile devices (such as Personal Digital Assistants (PDAs), cell phones, SmartCards, etc.) are being connected wirelessly either to check information or to carry out bussiness transactions. Information exchange among those devices and servers and/or personal computers by means of wireless networks has brought a real mobility. Nevertheless, wireless communication channels are even more unsecure than more traditional channels (mainly because being air is typical transmission media, it is relatively easy for an opponent to rob and/or modify information). Therefore it is strongly recommendable to put in place highly reliable and efficient security tools that allow information protection thus satisfying users security demands.

In this Dissertation, a system for secure data exchange within wireless environments with focus on mobile devices is presented. Our system is based on the client - Server paradigm, where the client is a mobile device. WAP protocol's Wireless Transport Layer Security (WTLS) was used for establishing secure sessions based on its *handshake* protocol. According to that protocol, cryptographic parameters are established for their later use throughout all information exchange sessions among participant entities. Quite particularly, a *session key* is generated by using any one of the two key crypto-systems supported by WTLS: Elliptic Curve Cryptosystems (ECC) and RSA.

On the other hand and in order to provide confidentiality during the information exchange phase, our system uses such session key to encrypt/decrypt data between client and server jointly with well-know symmetric cipher algorithms such as: DES, AES, TDES and so on. Likewise, our system offers the following security services: integrity and authentication using digital signature/verification and digital certificates tools.

Our results confirm that the Elliptic Curve Crypto-system paradigm is the most efficient option when dealing with implementations running on mobile device platforms which often have severe computational

power limitations. Indeed, we show that ECC gives similar security levels as the ones offered by RSA, but employing key sizes up to ten times smaller than the ones used by the later. Furthermore, our experiments performed on our information exchange system show that one needs to pay only a small computational overhead in order to execute either ECC or a hybrid scheme (which combines ECC and RSA) during the handshake phase; and symmetric cryptography during the data exchange phase.

Summarizing the main contributions of this Thesis work are:

- A WTLS handshake protocol implementation operating on wireless environments.
- The proposal of a scheme that combines ECC and RSA crypto-systems into a single hybrid scheme further improving the efficiency of the handshake protocol.
- A handshake protocol analytical model able to estimate the execution time associated with the handshake protocol.

Acrónimos

AC. Autoridad certificadora.

AES. Advanced Encryption Standard
(Estándar de cifrado avanzado).

AP. Access Point (Punto de acceso).

BSS. Basic Service Set
(Conjunto de servicios básicos)

CCMP. Counter-Mode with Cipher Block Chaining Message Authentication Code Protocolo
(Protocolo para autenticación de mensajes por encadenamiento de bloques cifrados)

CHAP. Challenge Handshake Authentication Protocol
(Protocolo de negociación para autenticación por retos).

CRC. Cyclic Redundancy Code
(Código de redundancia cíclica).

CSMA/CA. Carrie-Sense Multiple Access with Collision Avoidance
(Protocolo de acceso al medio).

CWML. Compact Wireless Markup Language
(Lenguaje de etiquetado inalámbrico compacto).

DES. Data Encryption Standard
(Estándar de cifrado de datos)

DS. Distribution System
(Sistema de distribución).

DSA. Data Signature Algorithm
(Algoritmo de firmado digital)

ECC. Elliptic Curve Cryptography
(Criptografía de curvas elípticas)

ECDSA. Elliptic Curve Data Signature Algorithm
(Algoritmo de firmado digital usando curvas elípticas).

EFF. Electronic Frontier Foundation.

ESS. Extended Service Set
(Conjunto de servicios extendido).

GSM. Global System for Mobile communication
(Sistema global para comunicaciones móviles).

IBSS. Independent Basic Service Set
(Conjunto de servicios básico independiente).

ICV. Integrity Check Value
(Valor de verificación de integridad).

IEEE. Institute of Electrical and Electronic Engineers
(Instituto de ingenieros electricos y electrónicos).

ISO/IEC. International Standards Organization/International Electrotechnical Commision
(Organización internacional de estándares / Comisión internacional de electrotécnicos).

ITU-T. International Telecommunication Union Telecommunication Standarization Sector
(Unión internacional de telecomunicaciones - Sector de estandarización de telecomunicaciones).

IV. Initialization Vector (Vector de inicialización).

LFSR. Linear Feedback Shift Register
(Registro de desplazamiento con retroalimentación lineal).

MAN. Metropolitan Area Network
(Red de área metropolitana).

MID. Módulo para Intercambio de Datos.

MN. Módulo de Negociación.

NIC. Tarjeta de Interfaz de Red.

NIST. National Institute for Standards and Technology
(Instituto nacional para estándares y tecnología).

OSA. Open System Authentication

(Autenticación de sistema abierto).

OSI.Open Systems Interconection
(Sistema abierto de interconexión).

PDA. Personal Digital Assistant
(Asistente personal digital).

PGP. Pretty Good Privacy
(Aplicación para intercambio de datos).

PKCS. Public Key Cryptography Standard
(Estándar de criptografía de llave pública).

SHA. Securete Hash Algorithm
(Algoritmo hash).

TDES. Triple DES.

TLS. Transport Layer Security.
(Capa de transporte seguro)

VPN. Virtual Private Network
(Red privada virtual)

WAP. Wireless Application Protocol
(Protocolo de Aplicaciones Inalámbricas)

WEP. Wired Equivalent Privacy.
(Protocolo de seguridad para redes inalámbricas).

WLAN. Wireless Local Area Networks
(Red de área local inalámbrica.)

WTLS. Wireless Transport Layer Security.
(Capa de transporte seguro inalámbrico)

Introducción

En la actualidad existe una gran demanda de conexión a redes inalámbricas por medio de dispositivos móviles, para obtener una serie de servicios, desde la predicción del clima y del tráfico hasta mantener una constante comunicación con la bolsa de valores, comprar boletos a eventos o pagar los impuestos vía Web, entre muchos más. El intercambio de información entre estos dispositivos y servidores o computadoras de escritorio, valiéndose de redes inalámbricas ha originado tener una movilidad real, lo cual ha dado paso al denominado Internet Móvil.

Las redes inalámbricas ofrecen la ventaja de la movilidad, pero sus características presentan ciertos inconvenientes en términos de seguridad con respecto a las redes tradicionales. Una de las desventajas es el medio de transmisión ya que las redes inalámbricas no pueden delimitar su señal, por lo que todo aquel dispositivo que esté dentro del rango de alcance de la señal emitida puede captar los datos y manipularlos. Otra desventaja es el ancho de banda, que aún es muy limitado, por lo que es muy costosa la transmisión de datos cuando se establece una sesión segura, así que se debe evitar enviar demasiada información. El uso de redes inalámbricas incrementa la posibilidad de intercambiar información, pero también aumenta la posibilidad de ataques a la misma, por lo que el proteger los datos se ha vuelto una necesidad prioritaria.

A pesar que existen protocolos y mecanismos de seguridad para redes inalámbricas (por ejemplo WEP), se ha demostrado que muchos de ellos tienen debilidades y carencias. Otro problema es que al no tener homogeneidad entre los distintos dispositivos móviles, el desarrollo de sistemas de seguridad debe ser casi a la medida porque el poder de cómputo y la limitante de la memoria varían de un dispositivo a otro. Las restricciones computacionales de los dispositivos móviles aún son un impedimento para poder echar mano a los sistemas de seguridad ya establecidos, por lo tanto, es necesario realizar aplicaciones altamente eficientes pero con una complejidad computacional baja para que puedan ser instaladas en dichos dispositivos. Una opción muy viable para proveer seguridad es hacer uso de la criptografía aprovechando los servicios de

seguridad intrínsecos que otorga.

Con el objetivo de entablar una sesión segura para el intercambio de información, en esta tesis se hace uso del protocolo de negociación del estándar TLS/WTLS utilizando los criptosistemas RSA, ECC y, adicionalmente, de un esquema híbrido que consiste en una combinación de ambos. En este trabajo de tesis se ha desarrollado un prototipo para intercambiar datos de forma segura en ambientes inalámbricos empleando dispositivos móviles. El sistema establece una sesión segura siguiendo las especificaciones del protocolo de negociacón de WTLS y posteriormente intercambiar información de forma confidencial usando criptografía de llave secreta. Asimismo, con el sistema se pueden firmar/verificar digitalmente documentos utilizando criptografía de llave pública. Uno de los objetivos de este trabajo es presentar un estudio del desempeño del protocolo de negociación en los dispositivos móviles. Además se diseñó un modelo analítico con el cual se obtuvieron tiempos estimados de duración del protocolo, los cuales fueron contrastados y corroborados experimentalmente.

Los resultados experimentales obtenidos en las pruebas realizadas corroboran que ECC y el modelo híbrido son las opciones criptográficas más idóneas para la implementación del protocolos de negociación del tipo WTLS/TLS en dispositivos móviles con restricciones computacionales.

El resto de esta tesis está organizado de la siguiente manera: en el capítulo 1 se describen los conceptos básicos relacionados con la tecnología inalámbrica y la criptografía, así como una breve descripción del sistema desarrollado y el objetivo del mismo. En el capítulo 2 se discute el servicio de confidencialidad y la manera de desarrollarlo. En el capítulo 3 se muestran los conceptos relacionados con el servicio de autenticación y se detallan los algoritmos existentes para proveer dicho servicio. El capítulo 4 aborda algunos de los protocolos de seguridad usados en la actualidad, poniendo especial atención al protocolo WTLS, asimismo, se describe un modelo analítico para obtener una estimación del tiempo de duración del protocolo de negociación de WTLS. En el capítulo 5 se describe el diseño, la arquitectura, características de componentes y detalles de la implementación del Sistema de Seguridad para el intercambio de datos en Dispositivos Móviles. En el capítulo 6 se reportan los resultados experimentales obtenidos del análisis de desempeño al sistema. Finalmente se dan las conclusiones obtenidas en este trabajo y se describe el trabajo futuro.

Capítulo 1

Conceptos Básicos

El almacenamiento e intercambio de información entre estos dispositivos y computadoras de escritorio valiéndose de redes inalámbricas ha originado tener una movilidad real. Dado que el valor de esta información varía con respecto a cada usuario, es necesario contar con herramientas eficientes para protegerla, tanto en el dispositivo móvil como cuando se transmite en una red inalámbrica.

En este capítulo se explican algunos conceptos básicos tanto de las redes inalámbricas como de criptografía. En la primera sección se presenta un panorama general de las redes inalámbricas, haciendo enfasis en su tecnología y estructura. También se discute el por qué es importante la seguridad en las redes inalámbricas y se mencionan algunos dispositivos móviles ligeros. En la sección 1.2 se presenta una introducción a la criptografía, es decir, los servicios de seguridad, su clasificación, las ventajas y desventajas de la criptografía de llave secreta y de la criptografía de llave pública y un panorama general de la autenticación básica. Finalmente se muestra una breve descripción del sistema desarrollado en este tema de tesis.

1.1. Tecnología inalámbrica

El término "inalámbrico" es usado para describir telecomunicaciones en las cuales las ondas electromagnéticas transportan la señal en parte o toda la ruta de comunicación. En los últimos años se ha producido un gran crecimiento en el desarrollo de comunicaciones inalámbricas y de dispositivos móviles. El objetivo principal de las redes inalámbricas es proporcionar conectividad y acceso a las redes cableadas, como si fuera una extensión de éstas últimas, pero con la ventaja de la movilidad que ofrecen las comunicaciones

inalámbricas. Tal es la popularidad de estas redes que los fabricantes de computadoras están integrando dispositivos para acceso a Redes de Área Local Inalámbricas (WLAN, por sus siglas en inglés) en sus equipos; un claro ejemplo es Intel [5], que fabrica el chipset Centrino para computadoras portátiles. En el año de 1997 el Institute of Electrical and Electronics Engineers (IEEE) dió a conocer el estándar IEEE 802.11, en este estándar se encuentran las especificaciones tanto físicas (PHY) como a nivel MAC que hay que considerar para implementar una red de área local inalámbrica. Otro de los estándares definidos y que trabajan en este mismo sentido es el ETSI HIPERLAN [4].

Las redes inalámbricas pueden categorizarse en:

- Redes de área amplia o metropolitana (WAN/MAN por sus siglas en inglés). Son las redes cuya cobertura es de miles de kilómetros; la tecnología que utilizan estas redes son las usadas para telefonía celular. Estas redes son conocidas como redes celulares (por ejemplo, GSM).
- Las WLAN son redes cuyo alcance es de centenas de metros. En este tipo de redes es en la que nos concentraremos en este tema de tesis.
- Las redes personales (PAN, por sus siglas en inglés) cubren distancias cortas y cerradas, ejemplos de tecnologías utilizadas son Bluetooth y la norma 802.15[8].

En esta sección describiremos las principales características de la tecnología inalámbrica del estándar 802.11.

1.1.1. Tecnología y estructura

En 1997 el IEEE adoptó el estándar IEEE 802.11, como el primer estándar para redes LAN inalámbricas (WLAN). Este estándar define el control de acceso al medio (MAC, por sus siglas en inglés) y las capas físicas para una red WLAN. El estándar es similar al estándar IEEE 802.3 Ethernet y, específicamente, considera los siguientes aspectos:

- Las funciones requeridas por un dispositivo 802.11 para poder operar punto a punto o integrarse a una red LAN convencional (conectada mediante cables).
- La operación del dispositivo 802.11 dentro de posibles traslapes de redes WLAN y la movilidad de este dispositivo dentro de múltiples redes de éste tipo.

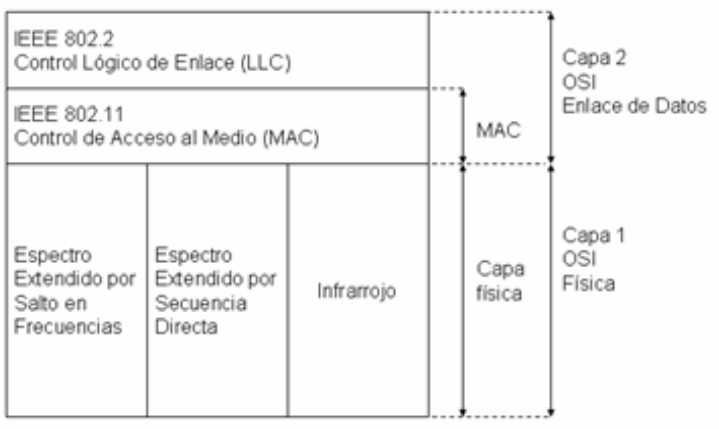

Figura 1.1: Mapeo entre el estándar 802.11 y el modelo de referencia OSI.

- El control de acceso al medio y los servicios de entrega de datos.
- Privacidad y seguridad de los datos que el usuario transmite sobre el medio inalámbrico (aire).

La figura 1.1 muestra el mapeo entre el estándar IEEE 802.11 y el modelo de referencia OSI.

Para propósitos de compatibilidad, la capa MAC 802.11 debe aparecer en las capas superiores de la red como un estándar 802 de redes LAN. La capa MAC 802.11 debe manejar la movilidad de los dispositivos de manera transparente a las capas superiores de la red LAN 802.

La arquitectura del estándar IEEE 802.11 está integrada por diversos componentes y servicios, los cuales interactúan para proveer la movilidad a los dispositivos. Entre los componentes y servicios del IEEE 802.11 se pueden mencionar a las estaciones inalámbricas y al Conjunto de Servicios Básicos.

La estación inalámbrica es el componente más básico de una red inalámbrica. Una estación es un dispositivo que contiene la funcionalidad del protocolo 802.11, esto es, tiene la capa MAC, las capas físicas y una conexión a la red inalámbrica. Típicamente las funciones del 802.11 se implementan en el hardware y software de la tarjeta de interfaz de red (NIC).

El estándar IEEE 802.11 define el Conjunto de Servicios Básicos (BSS, por sus siglas en inglés) como el componente base de una WLAN 802.11. El BSS consiste de un grupo de estaciones.

Empleando los componentes y servicios mencionados, se pueden describir las diferentes topologías para redes WLAN que existen.

Figura 1.2: Topología de conjunto de servicios básicos independientes

Conjunto de Servicios Básicos Independiente

La topología WLAN más básica está conformada por un conjunto de estaciones que se han reconocido entre sí y que están conectadas vía el medio inalámbrico de manera punto a punto. Esta forma de topología de red se conoce como Conjunto de Servicios Básico Independiente (IBSS) o redes ad-hoc. En una IBSS, las estaciones móviles se comunican directamente entre sí, sin embargo, cada estación móvil puede no ser capaz de comunicarse con otras debido a las limitaciones de la cobertura. En la figura 1.2 se muestra este tipo de topología.

Conjunto de Servicios Básicos de Infraestructura

Un Conjunto de Servicios Básicos de Infraestructura es un BSS con un componente denomidado punto de acceso (AP, por sus siglas en inglés). El punto de acceso provee la función de reenvío local para el BSS. Todas las estaciones en el BSS se comunican con el punto de acceso y no directamente. Todas las tramas son reenviadas entre las estaciones por el punto de acceso.

El punto de acceso también puede proveer conexión a sistemas de distribución.

La figura 1.3 muestra la topología descrita.

El Sistema de Distribución (DS, por sus siglas en inglés) es el medio por el cual distintos puntos de acceso se comunican entre sí con el propósito de intercambiar tramas entre las estaciones de sus respectivos BSSs, reenviar tramas para seguir a una estación móvil - ya que éstas se pueden mover de un BSS a otro - e intercambiar tramas con una red convencional.

El 802.11 extiende el rango de movilidad a un rango arbitrario a través del Conjunto de Servicios Extendidos (ESS). Un ESS es un conjunto de BSSs, donde los puntos de acceso se comunican entre ellos para

Figura 1.3: Topología de conjunto de servicios básicos de infraestructura

reenviar tráfico de un BSS a otro, a fin de facilitar el movimiento de estaciones entre BSSs.

La estructura de un ESS se muestra en la figura 1.4.

La capa MAC del 802.11 provee funcionalidad para permitir la entrega confiable de los datos sobre el medio inalámbrico. Sin embargo, no hay garantía de que las tramas serán entregadas con éxito debido a que el envío mismo se basa en una entrega sin conexión de la capa MAC. Asimismo, proporciona un método de acceso controlado al medio inalámbrico compartido llamado Carrier-Sense Multiple Access con Collision Avoidance (CSMA/CA).

Otra de las funciones de la capa MAC es proteger los datos que se van a transmitir, para esto cuenta con los servicios de seguridad y privacidad. La seguridad se provee por medio de los servicios de autenticación y por el protocolo Wireless Equivalent Privacy (WEP), el cual es un servicio de cifrado para los datos que se van a transmitir por la WLAN que será explicado en detalle en la sección 4.3.3.

La capa física del 802.11 es la interfaz entre la capa MAC y el medio inalámbrico por el cual las tramas se transmiten y reciben. La capa física provee tres funciones. Primero, proporciona una interfaz para intercambiar tramas con las capas superiores de la capa MAC para transmisión y recepción de datos. Segundo, usa la portación de señal y la modulación de espectro extendido para transmitir las tramas de datos sobre el medio. Tercero, provee una indicación de portación de señal de regreso a la capa MAC para verificar la actividad del medio.

Hoy en día, el IEEE ha públicado varias normas diferentes para las WLAN, entre los que destacan:

- 802.11b: Publicado en 1999, es una extensión al estándar 802.11 de 1997. La norma 802.11b permite lograr una velocidad teórica de transferencia de datos de 11 Mbps (similar al de una red Ethernet

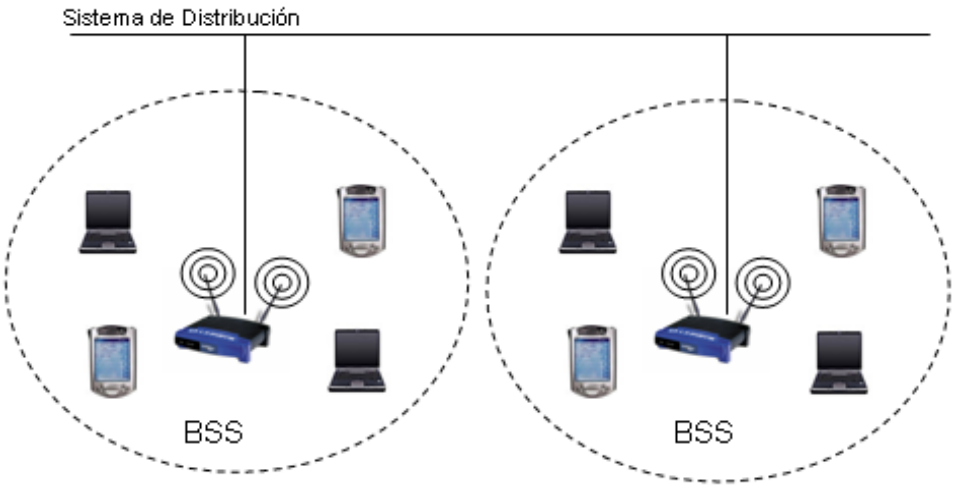

Figura 1.4: Estructura de un ESS

convencional). En la práctica, se logran velocidades entre 2 y 5 Mbps, lo que depende del número de usuarios, de la distancia entre emisor y receptor, de los obstáculos y de la interferencia causada por otros dispositivos. La interferencia es uno de los factores que más influye, porque los equipos 802.11b operan en la banda de 2.4 GHz, en la que se presenta interferencia de equipos como teléfonos celulares y hornos de microondas. Aún así, el estándar 802.11b se ha convertido en la variante más popular de esta norma.

- 802.11a: Dado a conocer al mismo tiempo que el 802.11b, con la intención de constituir una norma enfocada a redes inalámbricas para uso empresarial, a diferencia del estándar 802.11b, cuyo objetivo está enfocado a proveer conectividad a redes caseras y de pequeños negocios. Ofrece velocidades teóricas de hasta 54 Mbps (típicamente 22 Mbps) y opera en la banda de 5 GHz. Las principales desventajas de esta norma son su elevado precio, el hecho de que la banda de 5 GHz esté regulada en algunos países y su menor cobertura ha hecho que los equipos 802.11a sean menos populares que los 802.11b.

- 802.11g: Surgió en 2003, como la evolución del estándar 802.11b. Ofrece velocidades teóricas hasta de 54 Mbps (típicamente de 22 Mbps) en la banda de 2.4 GHz, y es compatible con los equipos 802.11b, por lo cual ha tenido una gran aceptación. Se prevé que reemplace por completo al estándar

802.11b en un futuro no muy lejano.

- 802.11i.Es el nuevo estándar del IEEE para proporcionar seguridad en redes WLAN. Incluye el nuevo algoritmo de cifrado AES (Advanced Encryption Standard). Este aspecto es importante puesto que significa que dispositivos con restricciones de cómputo no podrán utilizarlo. Para asegurar la integridad y autenticidad de los mensajes, la norma utiliza Counter-Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP). El 802.11i incluirá soporte no sólo para el modo BSS sino también para el modo IBSS (redes ad-hoc) [9].

1.1.2. Seguridad en redes inalámbricas

Debido a que el medio inalámbrico es compartido, todo lo que se transmite o recibe sobre la red inalámbrica puede ser interceptado. La confidencialidad y la autenticación se deben considerar siempre que se desarrolla un sistema de red inalámbrica. El objetivo al agregar las características de seguridad es hacer al tráfico inalámbrico tan seguro como el tráfico de una red convencional.

El tener como medio de comunicación al aire ocasiona que cualquier persona que tenga un dispositivo móvil y esté dentro del rango de cobertura (100 metros a la redonda aproximadamente), puede conectarse a una WLAN sin ningún problema. Si se considera el escenario de dos compañías A y B y cada una tiene una red inalámbrica N_A y N_B respectivamente, cualquier empleado de la compañía A, puede acceder a la información de la compañía B y viceversa. A su vez, puede existir otra entidad que esté dentro del rango de alguna de las redes y pueda modificar, eliminar o vender información, entrar libremente a Internet, robar software, utilizar la red como punto de ataque a otras entidades o infectar los sistemas con virus. En la figura 1.5 se muestra un esquema de este problema de seguridad en las redes inalámbricas.

Para evitar los ataques a las redes inalámbricas es necesario realizar un buen diseño de las mismas y considerar los siguientes aspectos:

- Asignación de llaves: Cada usuario (cliente o punto de acceso) de una red WLAN recibe su propia identificación que ha sido asignada por el administrador de red al configurar la red inalámbrica.

- Direcciones MAC: La dirección MAC es un identificador único que se le asigna a cada adaptador de una red. Si dicha dirección se usa como identificador en una lista de acceso, los adaptadores de red no autorizados se rechazan automáticamente.

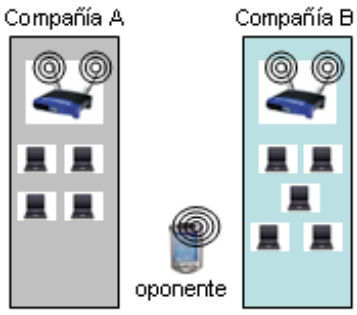

Figura 1.5: Ataques a redes inalámbricas

- Tecnología de red privada virtual (VPN): Las VPNs llevan ya bastante tiempo en operación y se consideran muy seguras. Se obtendrá una red local inalámbrica segura si se sabe sacar provecho de esta tecnología.
- Autenticación y codificación usando WEP: El estándar 802.11 implementa WEP como su tecnología de autenticación y codificación. El protocolo WEP se discutirá más detalladamente en el capítulo cuatro.

Los mecanismos actuales para proveer seguridad en ambientes inalámbricos aún tienen muchas carencias. Es necesario hacer uso de herramientas idóneas para realizar sistemas que otorguen a los dispositivos inalámbricos seguridad al momento de estar conectados a una red.

1.1.3. Dispositivos móviles ligeros

Se denominan dispositivos móviles a aquellos dispositivos que el usuario puede llevar consigo y que le permiten mantenerse comunicado, por ejemplo, computadoras de bolsillo, celulares, laptops. Sin embargo, debemos hacer una diferenciación entre estos dispositivos ya que, debido a su capacidad de cómputo y recursos, nos permiten realizar diferentes tareas. Principalmente clasificamos a los dispositivos móviles en ligeros y pesados. Un ejemplo de dispositivos móviles pesados es una laptop, la razón para categorizar a una computadora portátil como dispositivo pesado es porque su capacidad de cómputo es igual o superior a una computadora convencional.

Entre los dispositivos móviles ligeros podemos citar los siguientes:

- **Teléfonos Celulares.**

- **Teléfonos I-Mode**: En este tipo de teléfonos no se usa el protocolo WAP y se emplea una versión simplificada de HTML Asimismo, se utiliza el Lenguaje de Etiquetado Inalámbrico Compacto (CWML, por sus siglas en inglés) en lugar del Lenguaje de Etiquetado Inalámbrico de WAP.

- **Pagers**: Son dispositivos de telecomunicación pequeños que reciben (y algunas veces transmiten) señales y/o mensajes cortos.

- **PDAs**: Es un dispositivo que combina la computación con teléfono/fax, Internet y características de red. Un PDA típico puede funcionar como teléfono celular, emisor de fax, navegador web y organizador personal.

1.2. Criptografía

Los inicios de la criptografía se remontan a miles de años atrás pero para la década de los años 60 la proliferación de computadoras y de sistemas de comunicación fomentaron la investigación enfocada a la seguridad y como consecuencia a la criptografía. Una forma muy simple de definir criptografía es como la ciencia de ocultar mensajes de forma segura [2]. Sin embargo, las demandas actuales de seguridad de la información exigen más servicios, tales como, confidencialidad, integridad de datos, autenticación y no repudio. En las secciones siguientes se dará un panorama sobre los servicios de seguridad y sobre la clasificación de la criptografía.

1.2.1. Servicios de seguridad

En todos los esquema de comunicación existen amenazas tales como, interrupción, intercepción, generación o modificación de la información que se transmite o contra la identidad de los participantes. Para contrarrestar tales amenazas se definen una serie de servicios para proteger la información y minimizar cualquier ataque. Los objetivos de la seguridad se listan en la tabla 1.1 [1].

Confidencialidad	Mantener la información en secreto y solo autorizar a ciertas entidades a verla
Integridad de datos	Asegurar que la información no ha sido alterada por entidades no autorizadas
Autenticación o identificación	Corroborar la identidad de una entidad
Autenticación de mensajes	Corroborar la fuente de la información; también conocido como autenticación del origen de datos
Firmas	Mecanismo para ligar información con una entidad
Autorización	Permitir a una entidad realizar algo sobre información o recursos
Validación	Proporcionar periodos autorización para utilizar o manipular información o recursos
Control de Acceso	Dar privilegios de acceso a recursos restringidos a ciertas entidades
Certificación	Endoso de información acerca de algo o alguien por una entidad en la cual se confia
Estampas de tiempo	Registro de la fecha de creación o de la existencia de la información
Atestiguar	Verificar la creación o la existencia de la información por una entidad que no es la creadora de dicha información
Recibo	Reconocimiento de que se ha recibido la información
Confirmación	Reconocimiento de que los servicios se han proporcionado
Propiedad	Proveer a una entidad el derecho legal de utilizar o de transferir un recurso a otras entidades
Anonimato	Encubrir la identidad de una entidad implicada en algunos procesos
No repudio	Prevención de la negación de una entidad sobre acciones realizadas
Revocación	Anular la certificación o la autorización a una entidad o documento

Tabla 1.1 Objetivos de la seguridad de la información

La información contenida en la tabla 1.1 se puede englobar en cuatro conceptos básicos que se denominan servicios de seguridad, estos servicios son:

1. *Confidencialidad*. La confidencialidad asegura que la información sensible sólo podrá ser consultada o manipulada por usuarios, entidades o procesos autorizados. Existen algoritmos matemáticos para proveer la confidencialidad haciendo ilegible la información.

2. *Integridad*. La integridad da la certeza de que la información no ha sido modificada por entidades no autorizadas para hacerlo. Dentro de las posibles modificaciones están la escritura, modificación o borrado de segmentos de datos.

3. *Autenticación*. La autenticación asegura que la identidad de los participantes es verdadera. La autenticación se logra verificando dichas identidades usando mecanismos como firmas digitales, certificados digitales o características biométricas. El no contar con este servicio compromete a los servicios antes mencionados dado que cualquier entidad o usuario no autorizado puede realizar algún ataque a la seguridad. En el caso de los usuarios se pueden evaluar tres aspectos para autenticarlos, el primer aspecto es verificar algo que el usuario *tiene*, por ejemplo una llave privada con la cual puede emitir firmas digitales; el segundo aspecto es poner a prueba al usuario sobre algo que *sabe*, esto es, pedirle una contraseña y, finalmente, el tercer aspecto es verificar algo que el usuario *es*, por ejemplo, analizar sus huellas dactilares o su retina.

4. *No repudio*. El no repudio ofrece protección a un usuario o entidad frente a que otro usuario niegue posteriormente que en realidad se realizó cierta transacción. Esta protección se efectúa por medio de una colección de evidencias irrefutables que permitirán la resolución de cualquier disputa.

1.2.2. Clasificación de la criptografía

Antes de hacer una clasificación de la criptografía es necesario definir algunos conceptos relevantes:

- **Texto en claro**: Es la información que se desea compartir con otra u otras entidades. Esta información aún no sufre ninguna transformación por lo que aún no goza de la confidencialidad.

- **Llave**: La llave es un número generado aleatoriamente que poseen tanto el emisor como el receptor. Es necesario que se mantenga en secreto.

- **Cifrador**: El cifrador consta de un algoritmo matemático cuyo objetivo es hacer una transformación a la información de entrada para que el resultado sea ilegible. Este algoritmo realiza dicha tranformación con base a la llave que sea utilizada.

- **Texto cifrado**: Es el resultado de pasar el texto en claro a través de un cifrador. Dicha información es ilegible y sólo puede ser recuperada haciendo pasar el texto cifrado por el mismo cifrador usando la llave correspondiente, no necesariamente es la misma llave que se empleó para cifrar.

Con los componentes descritos anteriormente se puede definir matemáticamente lo que es un criptosistema como sigue:

Definición 1.1 *Criptosistema. Un criptosistema es la quintupla (P, C, K, ε, δ) donde:*

P es el conjunto finito de los posibles textos en claro.

C es el conjunto finito de los posibles textos cifrados.

K es el espacio de llaves, es decir, el conjunto finito de todas las posibles llaves.

$\forall \kappa \in K \; \exists E_\kappa \in \varepsilon$ (regla de cifrado), $\exists D_\kappa \in \delta$ (regla de descifrado)

Cada $E_\kappa : P \rightarrow C y D_\kappa : C \rightarrow P$ son funciones tales que $\forall m \in P, D_\kappa(E_\kappa(m)) = m$

Existen dos categorías en los que se pueden clasificar los métodos de cifrado/descifrado y, por consiguiente, a la criptografía. Tales esquemas son: cifradores simétricos o criptografía de llave secreta y cifradores asimétricos o criptografía de llave pública.

Criptografía de Llave Secreta.

En el esquema de llave secreta se asume que las partes involucradas son las únicas entidades que poseen la llave secreta, la fortaleza de los algoritmos de llave simétrica radica principalmente en que la llave sólo es conocida por las entidades que desean transmitir información de manera confidencial. Cabe destacar que uno de los principales problemas con este esquema es la distribución de la llave.

Los tamaños de las llaves utilizadas para cifrar/descifrar van desde los 64 bits (aunque actualmente se necesitan al menos 80 bits para considerar a una llave realmente segura), hasta 256 bits [1]. El esquema de criptografía de llave secreta se muestra en la figura 1.6 .

Las ventajas y desventajas de la criptografía de llave secreta son:

Ventajas

- Los cifradores simétricos pueden diseñarse para cifrar grandes cantidades de información. Algunos cifradores pueden implementarse en hardware cifrando hasta varios gigabytes por segundo. Mientras que en el caso de las implementaciones en software la tasa de cifrado sólo es de cientos de megabits por segundo.

- Las llaves usadas en los cifradores simétricos son más pequeñas que las usadas en los asimétricos.

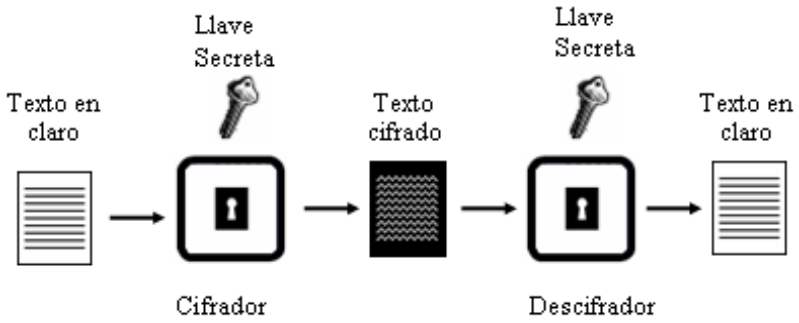

Figura 1.6: Esquema de cifrado de llave secreta

- Los cifradores simétricos pueden emplearse como primitivas para la construcción de varios mecanismos criptográficos incluyendo generadores de números pseudoaleatorios, funciones hash y esquemas eficientes de firma digital .
- Los cifradores simétricos pueden combinarse para producir cifradores más fuertes.

Desventajas

- En un esquema de comunicación entre dos entidades, la llave debe permanecer secreta en ambas partes.
- En una red grande, se necesita administrar y manejar muchas llaves. Por consecuencia, es necesario tener un buen esquema de administración de llaves. Se necesitan $\frac{(n)(n+1)}{2}$ llaves secretas en un sistema con n usuarios.
- En una comunicación entre dos entidades A y B, una práctica recomendada es cambiar frecuentemente la llave usada, donde lo ideal es que exista una llave para cada sesión que sostengan las entidades A y B.
- Los mecanismos de firma digital que se presentan en el cifrado simétrico requieren que las llaves sean grandes para la función pública de verificación o en dado caso el uso de una tercera entidad en la cual confíen las entidades involucradas en el esquema de firma digital.

Criptografía de Llave Pública.

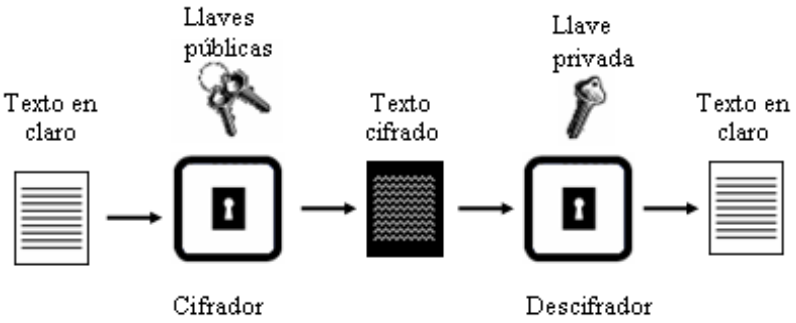

Figura 1.7: Esquema de cifrado de llave pública

En la figura 1.7 se muestra el esquema de criptografía de llave pública en donde cada una de las partes posee un par de llaves, una pública y una privada. La llave pública, como su nombre lo sugiere, se puede distribuir a cualquier entidad que desee tenerla, mientras que la llave privada sólo será conocida por el dueño de la misma. La llave pública será utilizada para cifrar la información y la llave privada para descifrar la misma o para el esquema de firma digital, la llave privada es usada para cifrar y la llave pública para descifrar..

Las ventajas y desventajas de la criptografía de llave pública son:

Ventajas.

- Solamente la llave privada debe estar guardada en secreto (sin embargo, la autenticidad de la llaves públicas debe estar garantizada).
- La administración de las llaves en una red requiere la presencia de una Tercera Entidad de Confianza y, dependiendo de las exigencias de la red, esta tercera entidad debe estar fuera de línea o en tiempo real.
- Dependiendo del modo de uso, las llaves privada y pública pueden durar por períodos de tiempo extensos, por ejemplo, varias sesiones o incluso varios años.
- Muchos esquemas de llave pública ofrecen mecanismos eficientes de firma digital. La llave usada para la función pública de verificación es mucho más pequeña que la usada por los esquemas de llave simétrica.

- En una red grande, el número de llaves necesarias puede ser considerablemente más pequeño que en un escenario simétrico.

Desventajas.

- El tiempo de ejecución de los métodos de cifrado de llave pública son considerablemente más lentos que los esquemas de llave simétrica.
- El tamaño de las llaves son mucho más grandes que las requeridas por los cifradores simétricos.
- El tamaño de la firma digital usando el esquema de llave pública es grande a comparación con las etiquetas de autenticación de datos que se proveen con las técnicas de llave simétrica.

1.2.3. Autenticación básica.

La autenticación es el mecanismo por el cual una entidad **A** se asegura de verificar que la identidad de una entidad **B** sea la que aquella dice ser y viceversa. Si este servicio no se cumple, cabe la posibilidad de que una entidad desconocida asuma una identidad que no le corresponde y con ello se compromete la privacidad y la integridad de la información que se intercambia.

Un esquema usado para proveer los servicios de autenticación y no repudio es el de firma digital. El propósito de una firma digital es proveer a una entidad un medio para enlazar su identidad a una pieza de información. En la figura 1.8 se pueden observar los procesos de firmado y verificación digital. El proceso de firma se hace con base en la utilización de un cifrador asimétrico, la diferencia que existe entre éste y un esquema de firma digital es que en este último se utiliza la llave privada para cifrar el resultado de la función hash o digesto, -donde una función hash transforma una cierta cantidad de información de tamaño variable en un bloque de tamaño específico -, y se usa la llave pública para descifrar el digesto. Los pasos para crear una firma digital son los siguientes:

1. El mensaje se hace pasar por una función hash para obtener su huella digital o digesto.
2. La huella digital se cifra utilizando la llave privada de la entidad **A**.
3. La firma digital se concatena con el mensaje original y es enviado a la entidad **B**.
4. La entidad **B** hace pasar el mensaje original por la misma función hash que haya utilizado la entidad **A**.

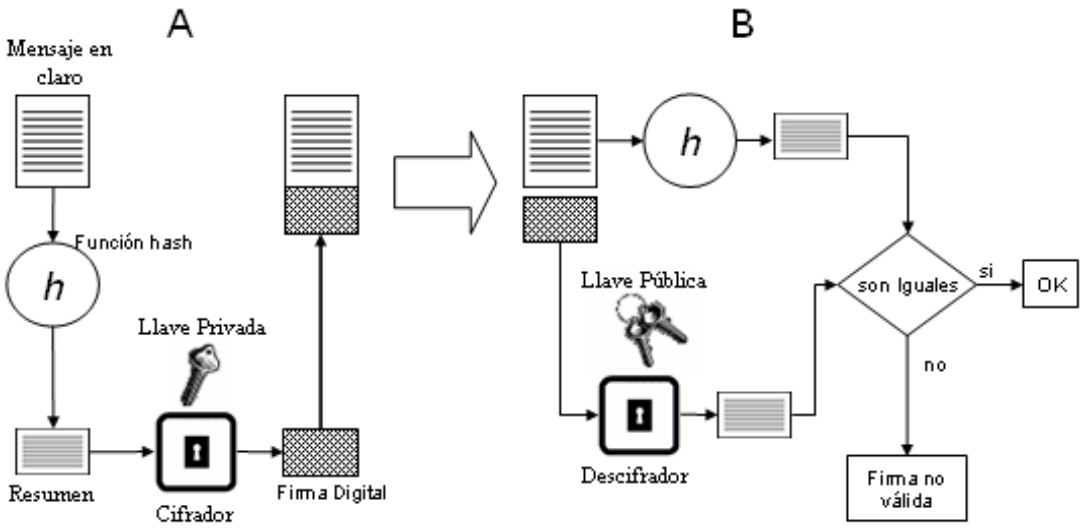

Figura 1.8: Esquema de firma/verificación digital

5. **B** toma la firma digital y la descifra utilizando la llave pública de la entidad **A**.

6. Las dos huellas digitales se comparan y si son idénticas es que la firma es válida (se ha autenticado a la entidad **A**) y el mensaje está íntegro (el mensaje original no sufrió ninguna alteración en la transmisión) .

Las ventajas y desventajas de las firmas digitales son:

Ventajas

- La primera y principal ventaja de la firma digital en comparación con la firma autógrafa, es que el procedimiento de verificación es exacto y que es imposible en la práctica su falsificación.

- La firma digital es portable, es decir, puede ser realizada en diferentes puntos del mundo, de forma simultánea y sin necesidad de testigos.

Desventajas

- La firma digital aún no es válida legalmente en muchos países.

- La seguridad de la firma digital depende de la llave privada, es decir, que si la llave privada se compromete por alguna causa, entonces, se compromete la seguridad de la firma digital, esto quiere decir que puede ser usada por individuos y entidades no autorizados.

En el resto de la tesis se explicarán con mayor presición cada uno de los esquemas de la criptografía.

1.3. Sistema de seguridad para intercambio de datos en dispositivos móviles.

El sistema diseñado en esta tesis está basado en el paradigma cliente - servidor, donde el cliente es un dispositivo móvil conectador inalámbricamente usando una red 802.11 ad-hoc. Se utilizó el protocolo WTLS, para establecer sesiones seguras con base a un protocolo de negociación con el cual se establecen los parámetros criptográficos que se emplearán a lo largo de la sesión de intercambio de información. Con estos parámetros se genera una llave de sesión, utilizando los criptosistemas de llave pública soportados por WTLS: Criptosistema de Curvas Elípticas y RSA, así como una implementación híbrida (RSA y ECC) que se anexo al protocolo de negociación, para mayor información sobre WTLS ver el Apéndice B. En la fase de intercambio de información el sistema utiliza dicha llave de sesión para cifrar/descifrar los datos entre el cliente y el servidor utilizando algoritmos tradicionales de cifrado simétrico tales como: DES, AES, TDES, para proveer confidencialidad. Asimismo, el sistema ofrece los siguientes servicios de seguridad: integridad y autenticación utilizando firma/verificación digital y certificados digitales.

El objetivo de esta tesis fue diseñar e implementar un sistema de seguridad para intercambio de datos utilizando el protocolo WTLS para crear sesiones seguras utilizando curvas elípticas para poder incrementar el nivel de seguridad y disminuir el uso de recursos en un dispositivo móvil. Así mismo, se utilizaron cifradores simétricos por bloques y cifradores por flujo de datos para proveer el servicio de confidencialidad. Los algoritmos hash que el sistema provee son SHA-1 y MD5 para la utilización en firmas digitales, expansión de contraseñas o verificación de certificados.

La figura muestra un diagrama general del sistema que se desarrolló. Como se puede observar en dicha figura, un cliente, es decir, un dispositivo móvil ligero inicia una petición de conexión con un servidor. Cuando dicha conexión se establece, el cliente propone una serie de parámetros criptográficos para establecer una negociación con el servidor y así crear una sesión segura. El Módulo de Negociación es el encargado de crear dicha sesión , es decir, crear una llave o secreto compartido utilizando el protocolo de WTLS. La llave de sesión se usa en el Módulo para Intercambio de Datos para proveer confidencialidad.

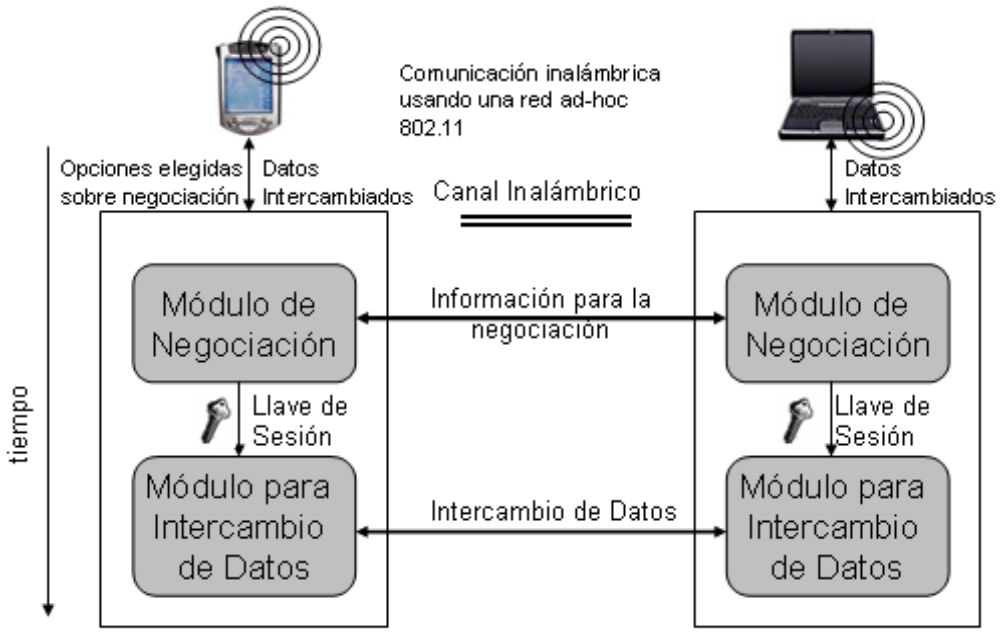

Figura 1.9: Diagrama general del sistema de seguridad para intercambio de datos en dispositivos móviles.

Capítulo 2

Servicio de Confidencialidad y su Implementación

Para la década de los años 60, la proliferación de computadoras y sistemas de comunicación fomentaron la investigación enfocada a la seguridad y, como consecuencia, a la criptografía. Siguiendo esa línea, uno de los primeros esquemas criptográficos desarrollados fue el de llave simétrica o secreta, en el cual se usa la misma llave para cifrar y descifrar la información enviada a través de un canal de comunicación. Dependiendo de cómo se lleva a cabo el procesamiento de la información es como se clasifican los criptosistemas de llave secreta. Así, se puede hablar de cifradores por bloque y por flujo. Ejemplos de cifradores por bloque son el Data Encryption Standard (DES) y el Advanced Encryption Standard (AES). Los cifradores por flujo de datos son cada vez más utilizados para proveer confidencialidad en dispositivos con restricciones computacionales y telefonía celular.

En este capítulo se describen algunos de los principales algoritmos tanto de la criptografía de llave secreta como de la criptografía de llave pública. En la sección 2.1 se describe una clasificación general de los cifradores simétricos, en la sección 2.2 se presentan los cifradores por bloques y la descripción de los algoritmos DES, Triple DES y AES. En la sección 2.3 se habla de los cifradores por flujo de datos y la descripción de los algoritmos A5 y RC4 y una de las aplicaciones de RC4, es decir, WEP. En la sección 2.4 se presenta la criptografía de llave pública y dos de los esquemas más usados en la actualidad, es decir, RSA y Criptografía de Curvas Elípticas

2.1. Clasificación de los cifradores simétricos

La criptografía simétrica hace una clasificación de los cifradores con base al tratamiento que se le da al mensaje que se cifra. Así, los cifradores simétricos pueden clasificarse en Cifradores por Bloques y Cifradores por Flujo de Datos.

A continuación se presentan algunas definiciones necesarias para el resto del capítulo [1].

Definición 2.1 *Sea K el espacio de llaves para un conjunto de transformaciones de cifrado. Una secuencia de simbolos* $e_1e_2e_3...e_i \in K$*, es llamada llave.*

Definición 2.2 *Sea A un alfabeto de q símbolos y sea* E_e *un cifrador simple de sustitución con bloques de longitud 1 donde* $e \in K$*. Tomando* $m_1m_2m_3...$ *como una cadena de texto en claro y tomando* $e_1e_2e_3...$ *como una llave perteneciente a K. Un cifrador por flujo de datos toma la cadena de texto en claro y produce una cadena de texto cifrado* $c_1c_2c_3....$ *donde* $c_i = E_{e_i}(m_i)$*. Si* d_i *denota la inversa de* e_i*, entonces* $D_{d_i}(c_i) = m_i$ *descifra la cadena de texto cifrado. Un cifrador por bloques divide el texto en claro que se va a transmitir en fragmentos de longitud t sobre un alfabeto A, cifrando un bloque a la vez.*

En las siguientes secciones se describen los dos tipos de cifradores, es decir, los cifradores por bloques y los cifradores por flujo de datos.

2.2. Cifradores por bloques

Un cifrador por bloques es un esquema de cifrado que divide el mensaje original o texto en claro en pequeños bloques de longitud *n* sobre un alfabeto *A*, cifrando un bloque a la vez [1].

Dos de los conceptos más importantes en los cuales se basan diversos algoritmos de cifrado simétrico son la *confusión* y la *difusión*, los cuales son técnicas para ocultar la redundancia en un texto en claro y fueron propuestos, por vez primera, en la Teoría de Shannon [7]. Una descripción general de ambos conceptos se muestra ensegida.

- Confusión: Trata de ocultar la relación entre el texto en claro y el texto cifrado. Dicha relación se da a partir de la llave utilizada, dado que si tal relación no existiera no sería posible descifrar los mensajes. El mecanismo más simple de la confusión es la sustitución, la cual consiste en cambiar cada ocurrencia de un símbolo en el texto en claro por otro.

- Difusión: Diluye la redundancia del texto en claro repartiéndola a lo largo de todo el texto cifrado. Para lograr la difusión se puede hacer uso de transposiciones, es decir, cambiar de lugar elementos individuales del texto en claro.

Los cifradores por bloque ofrecen diversos grados de seguridad, determinados esencialmente por el tamaño en bits de la llave secreta y por el diseño de cada cifrador. Una de las características que hacen fuerte a un cifrador es la buena utilización de la confusión y la difusión. Un modelo que usa de manera eficiente estos dos conceptos es el modelo Feistel y muchos cifradores están basados en él [7][1]. Los cifradores que usan el modelo Feistel dividen el bloque de longitud *n* en dos mitades, las cuales se denominan *L* y *R*. El cifrado se define en base a iteraciones o rondas. La salida de cada ronda se usa como entrada a la siguiente, considerando la siguiente relación:

$$\begin{aligned} L_i &= R_{i-1} \quad \text{si } i < n \\ R_i &= L_{i-1} \oplus f(R_{i-1}, K_i) \qquad \text{si } i < n \\ L_n &= L_{n-1} \oplus f(R_{n-1},\ K_n) \\ R_n &= R_{n-1} \end{aligned}$$

El modelo de Feistel tiene la propiedad de que para descifrar basta con aplicar el mismo algoritmo con las K_i en orden inverso. Las K_i son las llaves de ronda obtenidas de la llave original usando un algoritmo de expansión. Es importante notar que el cifrado y descifrado son independientes de cómo sea la función *f*. Los cifradores que se basan en este modelo utilizan transformaciones de álgebra lineal en forma de corrimientos lógicos, operadores Booleanos a nivel bit y transformaciones no lineales que son implementadas con bloques de substitución de bits conocidos como cajas-*S*. Dichas cajas son los únicos bloques no lineales presentes en el modelo. Se acepta de manera general que la calidad en eficiencia y seguridad de un algoritmo cifrador depende del buen diseño de dichos bloques.

Ejemplos de cifradores por bloques son DES, Triple DES y AES. En las siguientes secciones se describen brevemente cada uno de ellos.

2.2.1. DES

Este cifrador está basado en el algoritmo LUCIFER, el cual fue desarrollado por IBM a principios de los setenta.

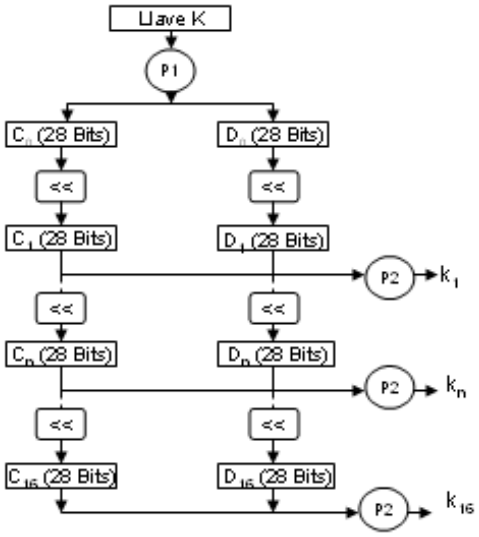

Figura 2.1: Creación de llaves de ronda

A mediados de 1998, se demostró que un ataque por fuerza bruta a DES era viable, debido a la escasa longitud que emplea en su llave. No obstante, el algoritmo aún no ha demostrado ninguna debilidad grave desde el punto de vista teórico.

El algoritmo DES codifica bloques de 64 bits empleando llaves de 56 bits. Se calcula un total de 16 valores de K_i, tal como se muestra en la figura 2.1, uno para cada ronda, efectuando primero una permutación inicial P1 sobre la llave de 64 bits, llevando a cabo desplazamientos a la izquierda de cada una de las dos mitades - de 28 bits- resultantes, y realizando finalmente una elección permutada (P2) de 48 bits en cada ronda, que será la K_i. Los desplazamientos a la izquierda son de dos bits, salvo para las rondas 1, 2, 9 y 16, en las que se desplaza sólo un bit. Debe notarse que aunque la llave para el algoritmo DES tiene en principio 64 bits, se ignoran ocho de ellos - un bit de paridad por cada byte de la llave, es decir, los bits 8, 16, 24, 32, 40, 56 y 64-, por lo que en la práctica se usan sólo 56 bits. Las tablas que describen a las permutaciones P1 y P2 se pueden consultar en [2].

El algoritmo se basa en una Red de Feistel de 16 rondas, más dos permutaciones, una que se aplica al principio ($P_{inicial}$) y otra que se aplica al final (P_{final}), tales que $P_{final} = P_{inicial}^{-1}$. Lo que hace la $P_{inicial}$ es que el bit 58 pasa a ser el primero, el bit 50 el segundo y así sucesivamente hasta tener el bit 7 como el último bit.

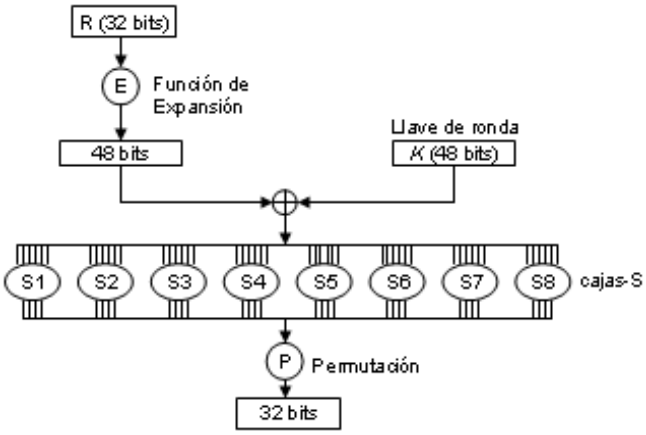

Figura 2.2: Esquema de la función f de DES

La función f que se muestra en la figura 2.2 se compone de una permutación de expansión (E), que convierte el bloque de 32 bits correspondiente en uno de 48. Posteriormente, se lleva a cabo una operación XOR con la llave de ronda K_i - también de 48 bits -. El resultado se fragmenta en pequeños bloques de 6 bits, los cuales se hacen pasar por cajas-*S* que producen 4 bits de salida, generando así un bloque de 32 bits. A estos 32 bits se les aplica una nueva permutación P. El funcionamiento de E está basado en una tabla de selección de bits [2], cuyo propósito es repetir algunos de éstos para generar un bloque de 48 bits.

La permutación que se realiza al final de la función f queda definida por una tabla que se puede consultar en [2].

Después de efectuar las 16 rondas siguiendo el algoritmo del modelo de Feistel, el bloque resultante se hace pasar por una permutación final (P_{final}), la cual es la inversa de la permutación inicial. La salida de P_{final} es un bloque de 64 bits de texto cifrado. Para descifrar, basta con usar el mismo algoritmo (ya que $P_{final} = P^{-1}_{inicial}$) empleando las llaves de ronda K_i en orden inverso.

2.2.2. Triple DES

A mediados de julio de 1998, una empresa llamada EFF (Electronic Frontier Foundation), fabricó una máquina capaz de descifrar un mensaje DES en menos de 22 horas. A pesar de una demostración tan clara de la falta de seguridad del algoritmo, DES sigue siendo ampliamente utilizado en multitud de aplicaciones.

El problema de DES no radica en su diseño, sino en que emplea una llave demasiado corta (56 bits),

lo cual hace que con el avance actual de las computadoras los ataques por fuerza bruta comiencen a ser opciones realistas. Mucha gente se resiste a abandonar este algoritmo, precisamente porque ha sido capaz de sobrevivir durante veinte años sin mostrar ninguna debilidad en su diseño, y prefieren proponer variantes que, por un lado evitarían el riesgo de tener que confiar en algoritmos nuevos, y por el otro permitirían aprovechar gran parte de las implementaciones por hardware existentes de DES. De todas la variantes que se han realizado una de la más usadas es Triple-DES.

Consiste en aplicar tres veces el algoritmo DES al mensaje original, con diferentes llaves. Si se consideran las expresiones $E_k(M)$ y $D_k(M)$ como las operaciones de cifrado y descifrado del mensaje M usando una llave K, respectivamente. Cada operación de cifrado para Triple-DES es una operación compuesta de las operaciones de cifrado y descifrado usando DES [13]. La ecuación que describe el cifrado para Triple-DES es la siguiente:

$$C = E_{k_1}(D_{k_2}(E_{k_1}(M)))$$

La ecuación que describe el descifrado para TDES es:

$$M = D_{k_1}(E_{k_2}(D_{k_1}(C)))$$

La interpretación de la expresión es la siguiente: Se codifica con la subllave k_1, se decodifica con k_2 y se vuelve a codificar con k_1. La llave resultante es la concatenación de k_1 y k_2, con una longitud de 112 bits.

El estándar especifíca tres opciones para elegir las llaves k_1, k_2 y k_3 las cuales son:

1. k_1, k_2 y k_3 son llaves independientes;
2. k_1 y k_2 son llaves independientes y $k_3 = k_1$;
3. $k_1 = k_2 = k_3$

Una aplicación tangible de Triple-DES se encuentra en los cajeros automáticos, los cuales lo tienen implementado como su algoritmo de cifrado.

2.2.3. AES

El National Institute for Standards and Technology (NIST) adoptó oficialmente al algoritmo Rijndael, en Octubre del año 2000, como nuevo Estándar Avanzado de Cifrado (AES) para aplicaciones criptográficas

no militares. La palabra Rijndael es un acrónimo formado por los nombres de sus dos autores, los belgas Joan Daemen y Vincent Rijmen.

AES es un cifrador por bloques, diseñado para manejar longitudes de llave y de bloque comprendidas entre los 128 y los 256 bits. Sin embargo, en el estándar adoptado por el gobierno estadounidense en noviembre de 2001 (FIPS PUB 197) [12], se especifica una longitud fija de bloque de 128 bits y la longitud de llave a escoger entre 128, 192 y 256 bits, aunque el tamaño de llave más utilizado es de 128 bits.

El algoritmo AES se basa en aplicar un número determinado de rondas a un valor intermedio que se denomina estado. Cada ronda es una composición de cuatro funciones invertibles diferentes, formando tres capas diseñadas para proporcionar resistencia frente a los ataques por criptoanálisis lineal y diferencial. A continuación se muestra el propósito de cada capa:

- La capa de mezcla lineal (funciones DesplazarFila y MezclarColumnas) permite obtener un alto nivel de difusión a lo largo de varias rondas.
- La capa no lineal (función ByteSub) consiste en la aplicación paralela de cajas-*S* con propiedades de no linealidad.
- La capa de adición de llave es llevada a cabo mediante una operación XOR entre el estado intermedio y la subllave correspondiente a cada ronda.

En cualquier parte del algoritmo, el estado de los datos procesados se puede representar mediante una matriz rectangular de bytes (conocida como matriz de estado), que posee cuatro filas, y N_b columnas. Donde N_b representa el número de palabras en el estado. Una palabra es un grupo de 32 bits que se trata como una sola entidad o como un arreglo de 4 bytes.

La llave K se representa mediante una matriz con cuatro filas y N_k columnas. La longitud de la llave es representada por $N_k = 4$, dicho número especifica el número de palabras.

El número de rondas que se llevan a cabo depende directamente de la longitud en palabras de la llave. El número de rondas es representado por N_r, donde $N_r = 10$ cuando $N_k = 4$, $N_r = 12$ cuando $N_k = 6$ y $N_r = 14$ cuando $N_k = 8$

El bloque que se pretende cifrar o descifrar se traslada directamente byte a byte sobre la matriz de estado, siguiendo la secuencia $a_{0,0}$, $a_{1,0}$, $a_{2,0}$, $a_{3,0}$, $a_{0,1}$ . . ., y análogamente, los bytes de la llave se copian sobre la matriz de llave en el mismo orden, a saber, $k_{0,0}$, $k_{1,0}$, $k_{2,0}$, $k_{3,0}$, $k_{0,1}$ . . ..

Siendo *B* el bloque que se quiere cifrar, y *S* la matriz de estado, el algoritmo AES con N_r rondas queda como sigue:

Calcular K_0,K_1, . . .K_{N_r} llaves de ronda a partir de la llave *K*.

S⟵ $B \oplus K_0$

Para *i = 1* hasta $N_r - 1$ hacer

S ⟵ *ByteSub(S)*

S ⟵ *DesplazarFila(S)*

S ⟵ *MezclarColumnas(S)*

S ⟵ $K_i \oplus S$ *(Adicionar la Llave de Ronda)*

fin de ciclo

S ⟵ *ByteSub(S)*

S ⟵ *DesplazarFila(S)*

S ⟵ $K_i \oplus S$ *(Adicionar la Llave de Ronda)*

Dado que cada ronda es una sucesión de funciones invertibles, el algoritmo de descifrado consiste en aplicar las inversas de cada una de las funciones en el orden contrario, y utilizar las mismas llaves de sesión K_i que en el cifrado, sólo que comenzando por la última. Enseguida se presentan las características de las funciones involucradas en cada ronda:

Función ByteSub

La transformación ByteSub es una sustitución no lineal que se aplica a cada byte de la matriz de estado, mediante una caja-*S* de 8*8 invertible, que se obtiene a través de la combinación de dos transformaciones, dichas transformaciones se pueden consultar en [12]:

En la figura 2.3 (a) se muestra la función ByteSub. La función inversa de ByteSub consiste en la aplicación de la inversa de la caja-*S* correspondiente a cada byte de la matriz de estado.

Función DesplazarFila

Esta transformación consiste en desplazar a la izquierda cíclicamente las últimas tres filas de la matriz de estado. Cada fila f_i se desplaza un número de posiciones c_i diferente. La primera fila no sufre ningún desplazamiento, el resto de valores viene en función de N_b. En la figura 2.3 (b) se puede observar el comportamiento de la función. La función inversa de DesplazarFila consiste en un desplazamiento de las filas de la matriz de estado a la derecha.

Función MezclarColumnas

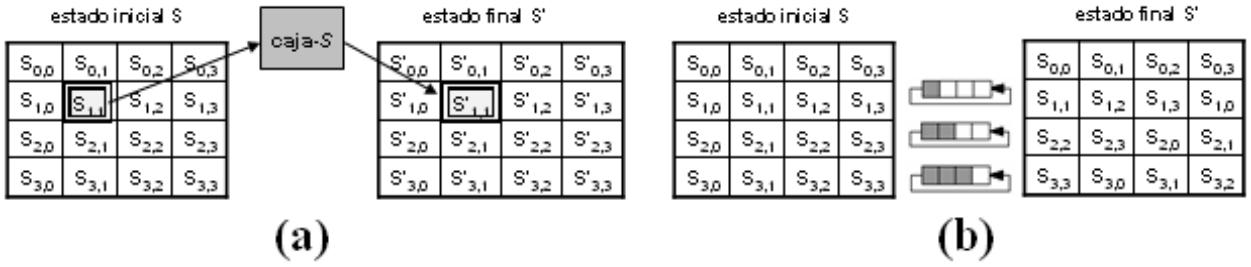

Figura 2.3: (a) Función de sustitución de bytes; (b) Función desplazar fila

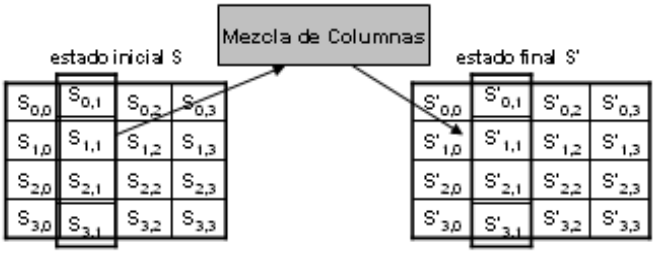

Figura 2.4: Función MezclarColumnas

Para esta función, cada columna de la matriz de estado se considera un polinomio cuyos coeficientes pertenecen a GF(2^8) y se multiplica módulo $x^4 + 1$ por:

$$c(x) = 03x^4 + 01x^2 + 01x + 02$$

donde 03 es el valor hexadecimal que se obtiene concatenando los coeficientes binarios del polinomio correspondiente en GF(2^8), en este caso 00000011, o sea, $x + 1$, y así sucesivamente. La figura 2.4 muestra la función MezclarColumnas.

La inversa de MezclarColumnas se obtiene multiplicando cada columna de la matriz de estado por el polinomio:

$$d(x) = 0Bx^4 + 0Dx^2 + 09x + 0E$$

Las diferentes llaves de ronda K_i se derivan de la llave principal *K* mediante el uso de dos funciones: una de expansión y otra de selección. Siendo *n* el número de rondas que se van a aplicar, la función de expansión

permite obtener, a partir del valor de *K*, una secuencia de $4 * (n + 1) * N_b$ bytes. La selección simplemente toma consecutivamente de la secuencia obtenida bloques del mismo tamaño que la matriz de estado, y los va asignando a cada K_i.

También se ha comprobado que es resistente a los criptoanálisis lineal y diferencial. El método más eficiente conocido hasta la fecha para recuperar la llave a partir de un par *texto cifrado-texto en claro* es la búsqueda exhaustiva, por lo que se puede considerar a este algoritmo uno de los más seguros en la actualidad.

2.3. Cifradores por flujo de datos

Los cifradores por flujo de datos forman una clase importante del esquema de criptografía de llave simétrica. En 1917, J. Mauborgne y G. Vernam inventaron un criptosistema perfecto [14]. Dicho sistema consistía en emplear una secuencia aleatoria de igual longitud que el mensaje, que se usaría una sola vez, combinándola mediante alguna función simple -usualmente el or exclusivo (XOR) -, con el texto en claro caracter a caracter. Sobra decir que el principal inconveniente de este esquema es que la llave debe de ser del mismo tamaño que el texto en claro. Por dicha razón el sistema de Vernam carece de utilidad práctica en la mayoría de los casos.

Como opción práctica se propuso crear un generador pseudoaleatorio que usara como semilla a la llave y así obtener cadenas de bits que sólo se usarían una vez para cifrar el mensaje empleando la función de XOR. Así, todo aquel que conozca la llave, es decir, la semilla, podrá reconstruir la secuencia pseudoaleatoria y será capaz de descifrar el mensaje.

Con la definición formal de un cifrador por flujo de datos, dada en la sección 2.1, y el antecedente de que usan generadores pseudoaleatorios se puede mencionar que existen dos tipos de Generadores de secuencia:

- Generadores síncronos: La secuencia es calculada de forma independiente tanto del texto en claro como del texto cifrado. Al utilizar un generador síncrono es necesario que el emisor y el receptor estén sincronizados para poder descifrar el texto. Como puede suponerse ésta es una desventaja de este tipo de generadores.

- Generador asíncrono o auto sincronizado: La secuencia es función de una semilla, más una cantidad fija de bits anteriores de la propia secuencia. Estos generadores son resistentes a la pérdida e inserción de información, ya que vuelven a sincronizarse de forma automática después de que llegan *n* bloques correctos de información.

Los cifradores por flujo de datos son más veloces que los cifradores por bloques porque combinan la secuencia generada con el texto en claro mediante una operación or exclusivo bit a bit, en lugar de dividir el mensaje en bloques para cifrarlos por separado. Cabe resaltar que estos criptosistemas no proporcionan una seguridad perfecta como en el caso del algoritmo de Vernam, porque al emplear un generador sólo se obtendrán tantas secuencias distintas como posibles valores iniciales de la semilla o llave.

2.3.1. A5

El algoritmo A5 se utiliza para cifrar el enlace entre un teléfono celular y la estación base en el sistema de telefonía móvil GSM (Global Systems for Mobile communications). Una conversación en el sistema GSM entre dos entidades **A** y **B** se transmite como una sucesión de tramas. Cada trama consta de 228 bits, de los cuales 114 se utilizan para la comunicación digitalizada de **A** a **B** y los otros 114 para la comunicación de **B** a **A**. Una nueva trama se envía cada 4.6 milisegundos.

El cifrador A5 produce los 228 bits pseudoaleatorios de cada trama que se sumarán bit a bit, mediante la operación XOR, con los 228 bits de conversación dando lugar a los 228 bits de conversación cifrada.

De acuerdo con [10], existen dos versiones de A5: la versión A5/1 o versión fuerte y la versión A5/2 o versión débil, la cual no se discutirá en esta tesis. La primera se utiliza principalmente en sistemas de telefonía móvil en países europeos mientras que la segunda se utiliza fundamentalmente fuera de Europa.

En las siguientes líneas se describe, de forma general, el funcionamiento de las dos versiones de A5.

A5/1 está basado en una combinación de tres registros de desplazamiento con realimentación lineal (LFSR, por sus siglas en inglés), donde un LFSR es un conjunto de *L* estados $\{S_0, S_1, ..., S_{L-1}\}$ capaces de almacenar un bit cada uno [7]. La tabla 2.1 muestra la descripción de cada registro.

Número de registro	Longitud en bits	Polinomio característico	Bit de reloj
1	19	$x^{19}+x^5+x^2+x+1$	8
2	22	$x^{22}+x+1$	10
3	23	$x^{23}+x^{15}+x^2+x+1$	10

Tabla 2.1 Descripción de los LFSR

La numeración de los bits inicia desde cero partiendo desde el bit menos significativo. Asimismo, posee un contador de trama de 22 bits.

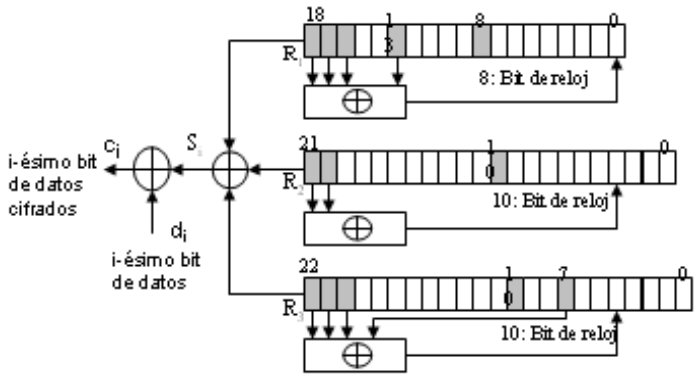

Figura 2.5: Esquema de cifrador A5/1

Los registros se desplazan de acuerdo a una función mayoría. Cada registro tiene asociado un bit de reloj. En cada ciclo, se examina este bit en los tres registros y se determina el bit que aparece más veces (bit mayoría). De esta forma, en cada paso se mueven dos o tres registros.

Al principio, los registros se inicializan con cero. Entonces en 64 ciclos de reloj, los 64 bits de la llave secreta se combinan de acuerdo al siguiente esquema: en los ciclos $0 \leqslant i \leqslant 64$, el bit $i-$ésimo de la llave se suma al bit menos significativo de cada registro mediante la operación XOR, es decir,

$$R[i] = R[i] \otimes K[i]$$

De manera similar, los 22 bits del contador de trama se suman en 22 ciclos. Posteriormente, el sistema completo se desplaza durante 100 ciclos en los cuales se descarta la salida. Después de que esta operación se ha completado, el cifrador está listo para producir los 228 bits de salida cifrados.

El esquema de esta versión de A5 se muestra en la figura 2.5.

2.3.2. RC4

RC4 es un cifrador por flujo desarrollado en 1987 por Ron Rivest para RSA Data Security, Inc. Es considerado secreto industrial y el algoritmo no se conoce abiertamente. Sin embargo, en septiembre de 1994 se publicó, de forma anónima, el código fuente en la lista de correo de Cypherpunks. Esta versión de RC4 rápidamente se difundió por todo el mundo y las personas con copias legales del mismo confirmaron

su compatibilidad.

A grandes rasgos, RC4 se puede describir de la siguiente manera:

1. A partir de una llave de longitud entre 1 y 256 bytes se inicializa una caja-*S* de 8 * 8.

2. La caja-*S* se usa para generar una lista de bytes pseudo-aleatorios cuyo tamaño coincide con el del texto a cifrar.

3. Estos bytes se combinan mediante una operación XOR con el texto en claro. El resultado es el texto cifrado.

De manera más detallada, se puede decir que RC4 tiene una caja-*S* de 8 * 8: $S_0,\ S_1,\ ...,\ S_{255}$ cuyas entradas son una permutación de los números 0 a 255. Además, tiene dos contadores, *i* y *j*, inicializados a cero.

Para cada byte de entrada se genera un byte pseudo-aleatorio con el cual se combina mediante la operación XOR dando lugar al byte cifrado.

Para generar un byte aleatorio se procede de la siguiente forma:

$$
\begin{aligned}
&i = (i + 1) \text{ mod } 256 \\
&j = (j + S_i) \text{ mod } 256 \\
&\text{Se intercambian } S_i \text{ y } S_j \\
&t = (S_i + S_j) \text{ mod } 256 \\
&K = S_t
\end{aligned}
$$

Se hace una operación XOR con el byte *K* y el texto en claro para producir el texto cifrado. Para recuperar el texto en claro se realiza una operación XOR con el texto cifrado y el byte *K*. El cifrado es rápido - cerca de 10 veces más rápido que DES [7].

Para inicializar la caja-*S* primero se llena linealmente, es decir, $S_0 = 0,\ S_1 = 1, ...,\ S_{255} = 255$. Después se llena otro arreglo de 256 bytes, copiando en él - 32 veces - los 8 bytes de la llave para cubrir todo el arreglo: $K_0,\ K_1, ...,\ K_{255}$. Se pone el índice *j* a cero. Entonces:

$$
\begin{aligned}
&\text{Desde } i = 0 \text{ hasta } 255 \\
&\quad j = (j + S_i + K_i) \text{ mod } 256 \\
&\quad \text{Se intercambian } S_i \text{ y } S_j
\end{aligned}
$$

Y eso es todo. El algoritmo es inmune a los criptoanálisis diferencial y lineal, no parece tener ningún ciclo pequeño y es altamente no lineal. El índice *i* asegura que cada elemento cambia y el índice *j* asegura

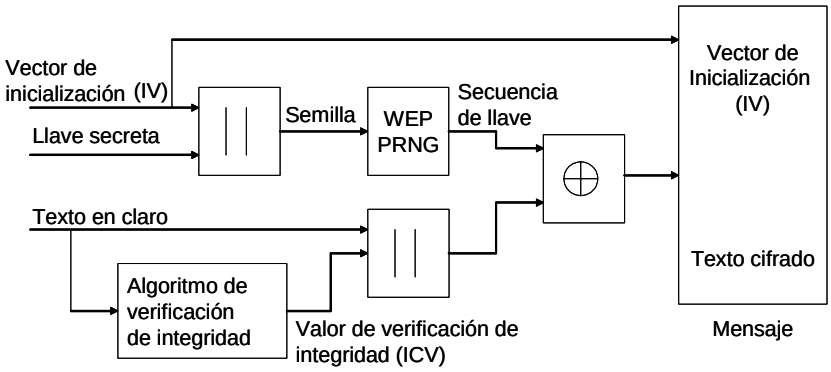

Figura 2.6: Proceso de cifrado en WEP

que los elementos cambian aleatoriamente.

Este algoritmo es muy utilizado en aplicaciones, por ejemplo WEP, el cual se describe a continuación.

2.3.3. WEP

En este protocolo se usa la misma llave para cifrar y descifrar los datos. La figura 2.6 muestra el funcionamiento del algoritmo de cifrado.

Como se puede apreciar en dicha figura, se aplican dos procesos al texto en claro, uno de éstos lo cifra y el otro lo protege contra modificaciones no autorizadas de los datos.

Para proteger los datos contra accesos no autorizados se utiliza el algoritmo de integridad de datos CRC-32 sobre el texto en claro, después de este proceso se obtiene el valor de verificación de integridad (ICV). El Código de Redundancia Cíclica, CRC, es un tipo de función hash que se usa para producir una suma de comprobación de un bloque de datos a fin de detectar errores en la transmisión. El CRC se calcula antes y después de la transmisión y se comparan ambos valores para confirmar que se trata del mismo. CRC es una familia de algoritmos y CRC-32 es un miembro de esta familia que produce 32 bits de salida (4 bytes).

La llave secreta, de 40 bits de longitud, se concatena con el vector de inicialización (IV) de 24 bits, lo cual da como resultado una llave de longitud total de 64 bits. Esta llave es la entrada a un generador de números pseudo-aleatorios, el cual genera una secuencia basada en la llave de entrada. La secuencia resultante se usa para cifrar los datos aplicando una operación XOR.

El mensaje que se envía es el vector de inicialización, el texto en claro y el valor de verificación de

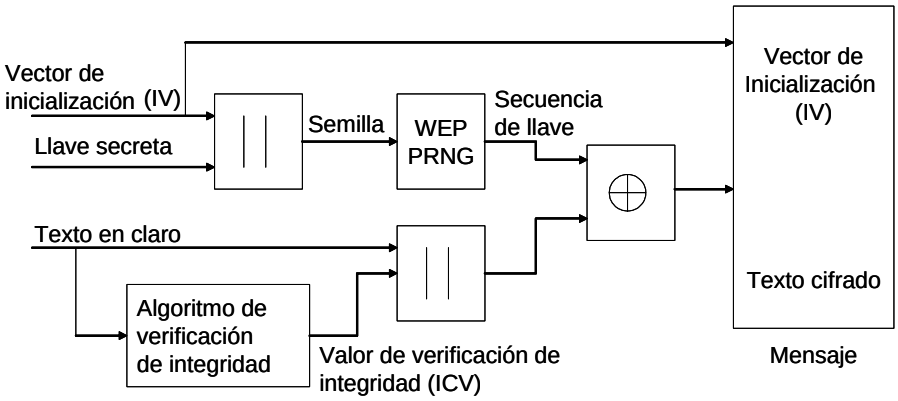

Figura 2.7: Proceso de descifrado en WEP

integridad.

Cuando se recibe el mensaje, el vector de inicialización que se envió se usa para generar la secuencia de llave necesaria para el proceso de descifrado. Combinando el texto cifrado con la secuencia de llave se obtiene el texto en claro original y el ICV. El descifrado se verifica aplicando al texto en claro recuperado el algoritmo CRC-32 y comparando la salida que produce con el ICV recibido. Si ambos valores difieren, significa que el mensaje se recibió con error y una indicación de esta situación se envía a la capa MAC de la estación emisora. Las estaciones móviles con mensajes erróneos no son autenticadas.

La figura 2.7 muestra el proceso de descifrado.

Existen dos tipos de autenticación:

- Autenticación de sistema abierto (OSA). La modalidad de autenticación OSA es vulnerable a los ataques debido al uso de llaves previamente compartidas, no utiliza criptografía de llave pública para crear una llave en un medio inseguro, de hecho, no se utiliza ningún protocolo de intercambio de llaves, por lo tanto, no hay autentificación verdadera.

- Autenticación de llave compartida. Provee un mejor grado de autenticación que el esquema de sistema abierto. La misma llave compartida usada para cifrar/descifrar las tramas de datos se emplea también para autenticar a la estación. La autenticación se lleva a cabo bajo un esquema de Autenticación por Retos (ver 4.3.1). Cuando el punto de acceso autentica a una terminal, lo que hace es asegurarse de que la terminal pertenezca a su grupo de dispositivos móviles. El punto de acceso no tiene forma de

determinar la identidad exacta de la terminal móvil que está requiriendo acceso. La mayoría de las implementaciones del 802.11 comparten las llaves a través de los puntos de acceso incrementando el tamaño del grupo en el cual un dispositivo móvil puede ser rastreado.

Problemas de WEP.

- Debilidad del vector de inicialización. La implementación del vector de inicialización (IV) en el algoritmo WEP tiene varios problemas de seguridad. Recordemos que el IV es la parte que varía de la llave (seed) para impedir que un posible atacante recopile suficiente información cifrada con una misma clave. La longitud de 24 bits para el IV forma parte del estándar y no puede cambiarse, por lo tanto, el número de IVs diferentes no es demasiado elevado ($2^{24} = 16$ millones aprox.), por lo que terminarán repitiéndose en cuestión de minutos u horas.

- Utilización de CRC-32. Entre los objetivos de WEP, se encuentra proporcionar un mecanismo que garantice la integridad de los mensajes. Por ello, WEP incluye un CRC-32 que viaja cifrado. Sin embargo, se ha demostrado [27] que este mecanismo no es válido y es posible modificar una parte del mensaje y a su vez el CRC, sin necesidad de conocer el resto. Esto permitiría, por ejemplo, modificar algún número de la trama sin que el destino se percatara de ello.

- WEP no incluye autentificación de usuarios. Cualquier usuario puede hacerse pasar por otra entidad y realizar cualquier tipo de ataque.

- La autenticación es de un solo sentido, es decir, se provee un mecanismo para que un punto de acceso autentifique a los dispositivos, pero no tiene ningún mecanismo para que los dispositivos autentiquen la red, es decir, al punto de acceso. Esto significa que un nodo impostor puede hacerse pasar por un punto de acceso y establecer comunicación con un dispositivo móvil. Dado que el dispositivo móvil no puede saber si se está comunicando con un punto de acceso auténtico, el nodo impostor tiene acceso a todo lo que se le envía.

2.4. Criptografía de llave pública

Cuando una entidad **A** desea comunicarse con una entidad **B** pero no han tenido un contacto previo en el cual hayan intercambiado la llave que les permitirá una comunicación segura, toda la información que **A**

envía a **B** puede ser interceptada por una tercera entidad, **C**. Sin embargo, es posible que **B** sea el único que pueda leer su información utilizando la *criptografía de llave pública*.

La criptografía de llave pública o asímetrica fue una contribución de Whitfield Diffie y Martin E. Hellman en 1976[3]. El esquema de cifrado asimétrico consta de un par de llaves, K_{pub} y K_{priv}, denominadas llave pública y llave privada, respectivamente, la figura 1.7 en la página 26, muestra el esquema de cifrado asimétrico. La llave pública se usa para cifrar la información, mientras que la llave privada se utiliza para descifrarla. Para el caso de las firmas digitales, la llave privada se usa para cifrar el digesto asociado al documento que se desea firmar y la llave pública se usa para descifrar dicho digesto y así llevar a cabo la verificación de la firma digital. Con los algoritmos de cifrado de llave pública se resuelve el principal problema del esquema de cifrado simétrico que es el intercambio de la llave, pero el costo computacional de los cifradores asimétricos es muy elevado. Un cifrador asimétrico se define de la siguiente manera [1]:

Definición 2.3 *Si se considera un esquema de cifrado que consiste en un conjunto de transformaciones de cifrado {E_e: e ∈ K} y descifrado {D_d: d ∈ K}. El método de cifrado se dice ser un esquema de cifrado de llave pública si para la asociación de cifrado/descifrado existe un par de llaves (e, d), la llave e (denominada llave pública), que está disponible públicamente, mientras la otra llave d (denominada llave privada), se mantiene en secreto. Para que el esquema sea seguro, debe de ser extremadamente difícil calcular d teniendo e.*

A continuación se presentan los algoritmos de criptografía de llave pública más importantes de la actualidad.

2.4.1. RSA

El algoritmo RSA, el cual debe su nombre a las iniciales del apellido de sus creadores: Ron Rivest, Adi Shamir y Leonard Adleman, fue creado en 1977 y utiliza la idea de llave pública. De forma general, está basado en el hecho de que la factorización de números primos grandes es un problema difícil.

El algoritmo funciona de la siguiente manera:

B escoge dos números primos grandes distintos, *p* y *q*. Realiza el producto de ambos para obtener:

$$n = pq$$

También escoge el exponente de cifrado *e* tal que:

$$gcd(e,\ (p-1)(q-1)) = 1$$

B envía el par (n, e) a **A** y mantiene los valores *p* y *q* en secreto.

A escribe su mensaje como un número *m*. Si *m* es más grande que *n* entonces divide el mensaje en bloques, cada uno menor que *n*. Por simplicidad, se asume que $m < n$. **A** calcula:

$$c \equiv m^e \ (mod\, n)$$

y envía *c* a **B**. Puesto que **B** conoce *p* y *q*, puede calcular $(p-1)(q-1)$ y, por lo tanto, puede encontrar el exponente de descifrado *d* con:

$$de \equiv 1 \ (mod\ (p-1)(q-1))$$

Como se muestra mas adelante,

$$m \equiv c^d \ (mod\, n)$$

de esta forma, **B** puede leer el mensaje.

En la tabla 3.1 se resume el algoritmo RSA.

1. **B** elige los números primos secretos *p* y *q* y calcula $n = pq$
2. **B** elige *e* con $gcd(e, (p-1)(q-1)) = 1$
3. **B** calcula *d* con $de \equiv 1 \ (mod\ (p-1)(q-1))$
4. **B** publica *n* y *e* y mantiene en secreto a *p*, *q* y *d*
5. **A** cifra *m* como $c \equiv m^e \ (mod\, n)$ y envía *c* a **B**
6. **B** descifra calculando $m \equiv c^d \ (mod\, n)$

Tabla 3.1 El algoritmo RSA

La razón por la que $m \equiv c^d \ (mod\, n)$ se debe al teorema de Euler que dice: Si $gcd(a,\ n) = 1,$ entonces $a^{\phi(n)} \equiv 1 \ (mod\, n)$. En este caso, $\phi(n) = \phi\,(pq) = (p-1)(q-1)$. Supóngase que $gcd(m, n) = 1$. Esto es probable dado que *p* y *q* son grandes, *m* seguramente no tiene a ninguno de los dos como factor. Dado que $de \equiv 1 \ (mod\ \phi(n))$ se puede escribir $de = 1 + k\phi(n)$ para algún entero k. Por lo tanto,

$$c^d \equiv (m^e)^d \equiv m^{1+k\phi(n)} \equiv m \cdot \left(m^{\phi(n)}\right)^k \equiv m \cdot 1^k \equiv m \ (mod\, n)$$

Y **B** puede recuperar el mensaje.

Existen diferentes ataques a RSA [2], aquí se describen dos de ellos.

Ataque a módulo común

Una posible implementación de RSA consiste en asignar el mismo módulo *n* a distintos usuarios pero diferentes valores para los exponentes *e* y *d*. Desafortunadamente, esto no funciona debido a que si el mismo mensaje se cifra con el mismo módulo y con dos diferentes exponentes que son primos entre sí, el texto en claro puede recuperarse sin ninguno de los exponentes privados.

Supóngase que *m* es es texto en claro, los exponentes públicos son e_1 y e_2 y el módulo común es *n*. Los dos textos cifrados son:

$$\begin{aligned} c_1 &= m^{e_1} \, mod \, n \\ c_2 &= m^{e_2} \, mod \, n \end{aligned}$$

El criptoanalista conoce *n*, $e_1,\ e_2,\ c_1$ y c_2 y con estos datos puede recuperar *m* porque dado que e_1 y e_2 son primos relativos, utilizando el algoritmo extendido de Euclides se pueden encontrar enteros *r* y *s* tales que:

$$re_1 + se_2 = 1$$

Suponiendo que *r* es negativo, entonces el algoritmo extendido de Euclides se utiliza nuevamente para calcular c_1^{-1}. Entonces:

$$\left(c_1^{-1}\right)^{-r} * c_2^s = m \, mod \, n$$

Ataque basado en un exponente público bajo

Aunque un exponente *e* bajo acelera el cifrado, también lo hace más inseguro. Si se cifran $e\,(e+1)\,/2$ mensajes linealmente independientes con diferentes llaves públicas y el mismo valor de *e* existe un ataque contra el sistema. Si los mensajes son idénticos entonces *e* mensajes son suficientes. La solución más sencilla a este problema es añadir a los mensajes valores aleatorios independientes. Esto también asegura que $m^e \, mod \, n \neq m^e$.

2.4.2. Criptografía de curvas elípticas

En la década de 1980, Miller y Koblitz introdujeron las curvas elípticas en la criptografía. Reciben su nombre no porque describan elipses, sino por su relación con las integrales elípticas [14]. Si bien su complejidad teórica es relativamente elevada, presenta algunas ventajas con respecto a otras técnicas como el algoritmo RSA. Entre estas ventajas se encuentra la eficiencia de las implementaciones y, sobre todo, que con llaves mucho más pequeñas se pueden alcanzar niveles equiparables de seguridad. Por ejemplo, se

estima que ciertos sistemas con llaves de 4096 bits se pueden reemplazar con sistemas de curvas elípticas de 313 bits.

Para explicar el cifrado y descifrado utilizando curvas elípticas es necesario mostrar un breve marco teórico de las mismas.

Una curva elíptica está definida por una ecuación con dos variables x,y con coeficientes. Para aplicaciones criptográficas, dichas variables y coeficientes deben ser elementos de un campo finito. Sobre el conjunto R, una curva elíptica es el conjunto de puntos del plano (x, y) que cumplen la siguiente ecuación:

$$y^2 = x^3 + ax + b$$

Los coeficientes a y b caracterizan unívocamente cada curva. Si $x^3 + ax + b$ no tiene raíces múltiples, entonces la curva correspondiente, en conjunción con un punto especial *O*, llamado **punto en el infinito**, más la operación suma, es lo que se denomina grupo de curva elíptica $E(R)$. *O* es un punto imaginario situado por encima del eje de las abscisas a una distancia infinita, y que por lo tanto no tiene un valor concreto.

Sumar un punto p consigo mismo k veces, es equivalente a multiplicar dicho punto por el escalar k, y se denota como kp.

La criptografía de curvas elípticas usa dos familias de éstas: Curvas primas definidas sobre Z_p y curvas binarias definidas sobre $GF(2^n)$. Las curvas primas son mejores para implementaciones de software, mientras que las curvas binarias son mejores para hardware [15].

Para las curvas elípticas sobre Z_p, se usa una ecuación cúbica en donde cada variable y coeficiente pertenecen al conjunto de los enteros entre 0 y $p - 1$, para cualquier número primo p; los cálculos son realizados módulo p. La ecuación característica para este tipo de curvas es:

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

Un campo finito $GF(2^m)$ consiste de 2^m elementos, junto con las operaciones de adición y multiplicación. Para las curvas elípticas sobre $GF(2^m)$, se usa una ecuación cúbica donde las variables y coeficientes pertenecen a $GF(2^m)$, para cualquier número m. La estructura de una ecuación cúbica apropiada para aplicaciones criptográficas es la siguiente:

$$y^2 + xy = x^3 + ax^2 + b$$

donde $x, y, a, b \in\ GF(2^m)$.

A manera de comparación, la operación de adición en criptografía de curvas elípticas es la contraparte de la multiplicación modular en RSA, y la adición múltiple es la contraparte de la exponenciación modular [15].

El problema en el cual se basa la criptografía de curvas elípticas es el problema de logaritmo discreto, el cual plantea lo siguiente: Si se considera la ecuación $Q = kP$, donde $Q, P \in E_p(a, b)$ y $k < P$. Es relativamente fácil calcular Q si se tiene k y P, pero es difícil calcular k teniendo Q y P.

Lo primero que hace **A** es codificar el mensaje para que sea enviado como un punto (x, y) llamado P_m. Cabe señalar que la selección de la coordenada (x, y) no es aleatoria ya que puede suceder que no esté contenida en el conjunto de puntos $E_p(a, b)$ que satisfagan a la ecuación de la curva que se este usando. Los métodos de codificación del mensaje pueden consultarse en [1].

A y **B** deben ponerse de acuerdo sobre ciertos parámetros, como es el caso de un punto G y un grupo elíptico $E_p(a, b)$. La entidad **A** selecciona una llave privada n_A y genera una llave pública $P_A = n_A \times G$.A su vez, **B** también gerera su par de llaves.

Para cifrar y enviar un mensaje P_m a **B**, **A** elige un número entero positivo aleatorio denominado k y produce el texto cifrado C_m el cual consiste del siguiente par de puntos.

$$C_m = \{kG, P_m + kP_B\}$$

Como se puede ver, para cifrar **A** enmascara el mensaje P_m sumando el punto resultante de multiplicar P_B por un número k; el cual sólo conoce **A**. A pesar de que P_B es la llave pública de **B**, nadie puede calcular el valor de k a partir de kP_B.

Para descifrar C_m, **B** multiplica el primer punto, kG, por su llave secreta y el resultado se resta al segundo punto $(P_m + kP_B)$. La siguiente ecuación muestra el descifrado:

$$P_m + kP_m - n_B(kG) = P_m + k(n_BG) - n_B(KG) = P_m$$

Para que un oponente recupere el mensaje es necesario calcular k teniendo k y kG, por lo que se enfrenta a un problema muy difícil.

Tanto RSA como ECC pueden cifrar mensajes de longitud arbitraria, pero como se discutió en el capítulo 1, una desventaja de estos cifradores es su tiempo de ejecución, por lo que la utilidad real de dichos cifradores es para intercambiar llaves secretas y para esquemas de firma digital. En la siguiente sección se presentan los algoritmos de firma digital que utiliza RSA y ECC. Además se presenta el algoritmo DSA.

Capítulo 3

Servicio de Autenticación Básica e Integridad de Datos

Las funciones hash permiten crear huellas digitales del mensaje para proveer el servicio de integridad de los datos. Sin embargo, el crear huellas digitales no es la única aplicaciones de las funciones hash, también se pueden generar llaves a partir de una contraseña o generar número pseudoaleatorios. Por otra parte, si dicha huella digital es firmada usando criptografía de llave pública se proveen los servicios de autenticación e integridad de datos simultáneamente.

En este capítulo se muestra un panorama de las funciones hash y de la firma/verificación digital, en la sección 3.1 se describe qué es una función hash, así como los algoritmos MD5 y SHA-1. Finalmente se muestran los algoritmos de Firma/Verificación digital PKCS, DSA y ECDSA.

3.1. Funciones hash

De manera general, una función hash transforma una cierta cantidad de información de tamaño variable en un bloque comúnmente llamado Resumen o Message Digest por su nombre en inglés. El digesto tiene un tamaño específico, típicamente de 128 o 160 bits, dependiendo del algoritmo que se use. Dicho resumen, es único e irrepetible, es decir, depende de la información que se suministra a la función hash. Si la información cambia, aún cuando el cambio sea mínimo, la salida de la función será diferente. Con las funciones hash se provee el servicio de integridad de datos. En la figura 3.1 se muestra el esquema de una función hash, la

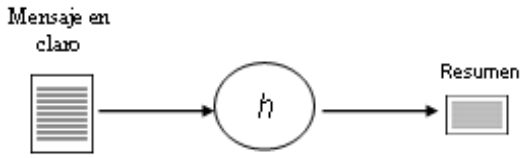

Figura 3.1: Esquema convencional de una función hash

cual, formalmente se define de la siguiente manera [1]:

Definición 3.1 *Una función hash es una función computacionalmente eficiente que mapea cadenas de bits de longitud arbitraria a cadenas binarias de longitud fija, llamadas en conjunto resumen".*

Uno de los objetivos de las funciones hash es que el resumen sirva como una huella digital de la información de entrada y pueda utilizarse como identificación única de dicha información. Puede darse el caso de que alguna función reciba como entrada un mensaje de longitud arbitraria y producen un bloque de longitud fija. Sin embargo, una función hash tiene, además, características que la hacen más eficiente y confiable, como las siguientes:

1. Dado el mensaje *M*, es fácil calcular *d = H(M)*, donde *d* es el resumen y *H* es la función hash .

2. La función hash debe de ser unidireccional o de solo ida, es decir, si se conoce *H(M)* encontrar *M* implica calcular todos los *M* posibles

3. La función hash debe resistir las colisiones, es decir, no debe ser posible encontrar un *M* y *M'* con $M \neq M'$ tal que *H(M) = H(M')*.

Cabe señalar que las anteriores características son ideales, ya que existen un espacio muy grande de mensajes, por lo tanto existe la posibilidad de encontrar un *M* y un *M'*diferentes que produzcan un *H(M)* igual, sin embargo la probabilidad de hallarlo debe ser infinitesimal.

En las siguientes secciones se muestra la descripción de dos de los algoritmos más importantes que existen en la actualidad para funciones hash: MD5 y SHA-1.

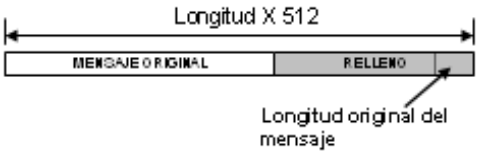

Figura 3.2: Proceso de rellenado del mensaje original

3.1.1. MD5

Este algoritmo fue desarrollado por Ronald Rivest en 1995 y es una versión mejorada de MD4, aunque es más complejo que éste, es similar en diseño y ambos producen un resumen de 128 bits a partir de un texto de cualquier longitud.

MD4 fue desarrollado para mejorar el rendimiento de MD2 , sin embargo, se detectaron diversos problemas y en 1996 se publicaron elementos que hacen hoy en día inservible el algoritmo. MD5 sustituyó a MD4 y aunque no tiene el rendimiento de su antecesor, hasta el momento no han sido publicados elementos que comprometan su integridad y funcionamiento [7].

El algoritmo de MD5 procesa el texto de entrada en bloques de 512 bits divididos en 16 sub-bloques de 32 bits cada uno. La salida es un conjunto de 4 bloques de 32 bits, que son concatenados para formar un único bloque de 128 bits.

MD5 comienza rellenando el mensaje de forma que la longitud del mismo sea 64 bits menos que un entero múltiplo de 512. El relleno consiste en un bit en 1 seguido por cuantos bits en 0 sean necesarios. La longitud original del mensaje es almacenada en los últimos 64 bits del relleno. En la figura 3.2 se puede apreciar lo antes dicho.

Adicionalmente, cuatro registros de 32 bits, llamados (*A*, *B*, *C*, *D*), se inicializan con los siguientes valores hexadecimales:

$$\begin{aligned} A &= 0x01234567 \\ B &= 0x89ABCDEF \\ C &= 0xFEDCBA98 \\ D &= 0x76543210 \end{aligned}$$

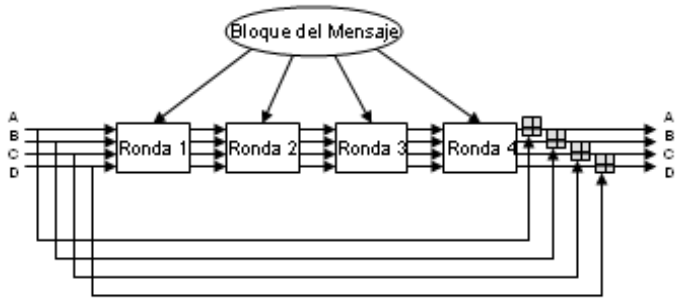

Figura 3.3: Ciclo principal de MD5

Los registros anteriores son copiados a cuatro variables diferentes $a = A$, $b = B$, $c = C$ *y* $d = D$.

Durante varias rondas de procesamiento el algoritmo toma bloques de 512 bits de la entrada y los mezcla con los cuatro registros. Este proceso se repite hasta que todos los bloques de entrada han sido consumidos. En la figura 3.3 se muestra el ciclo principal del algoritmo MD5. Como se puede observar en dicha figura, el ciclo principal tiene 4 rondas, todas muy similares. Cada ronda usa diferentes operaciones 16 veces. Cada operación realiza una función no lineal usando 3 de las variables *a, b, c* o *d* como entrada. Al resultado de la función no lineal se le suman el contenido de la variable que no se usó como entrada de la función lineal, un sub-bloque del mensaje (M_j) y una constante (t_i). Despues de sumar estos cuatro elementos, al resultado se le aplica un corrimiento circular de *S* bits a la izquierda, posteriormente se suma el contenido de una de las variables *a, b, c* o *d*. Finalmente, el resultado reemplaza el valor de una de las variables antes mencionadas. La figura 3.4 muestra el esquema de una operación en MD5.

Las cuatro funciones no lineales usadas en cada operación son las siguientes:

$$\begin{aligned} F(X,Y,Z) &= (X \wedge Y) \vee ((\neg X) \wedge Z) \\ G(X,Y,Z) &= (X \wedge Z) \vee (Y \wedge (\neg Z)) \\ H(X,Y,Z) &= X \oplus Y \oplus Z \\ I(X,Y,Z) &= Y \oplus (X \vee (\neg Z)) \end{aligned}$$

Cabe señalar que para cada ronda la función es diferente.

Si M_j representa el $j - ésimo$ sub-bloque del mensaje (desde 0 hasta 15), y $<<< S$ representa un

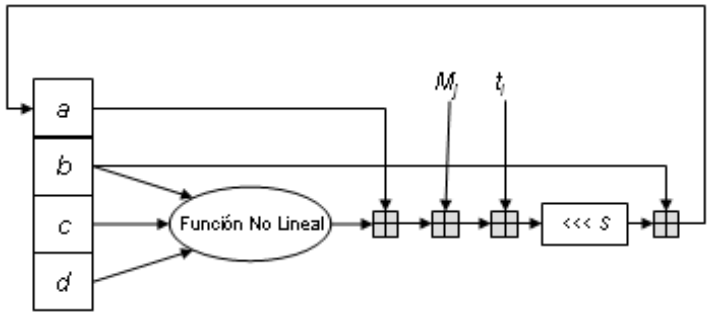

Figura 3.4: Una operación MD5

desplazamiento de bits circular a la izquierda, las cuatro operaciones están definidas como:

$$\begin{aligned} FF(a,b,c,d,M_j,S,t_i) \text{ representa } a &= b + ((a + F(b,c,d) + M_j + t_i) <<< S) \\ GG(a,b,c,d,M_j,S,t_i) \text{ representa } a &= b + ((a + G(b,c,d) + M_j + t_i) <<< S) \\ HH(a,b,c,d,M_j,S,t_i) \text{ representa } a &= b + ((a + H(b,c,d) + M_j + t_i) <<< S) \\ II(a,b,c,d,M_j,S,t_i) \text{ representa } a &= b + ((a + I(b,c,d) + M_j + t_i) <<< S) \end{aligned}$$

Al finalizar las cuatro rondas $A = A + a$, $B = B + b$, $C = C + c$ y $D = D + d$, y se continua con el siguiente bloque de datos. La salida final de MD5 es la concatenación de los registros *A*, *B*, *C* y *D*.

MD5 no es el único algoritmo de hash conocido. Por ejemplo, existe otra función llamada Secure Hash Algorithm, (SHA), desarrollado por la National Security Agency (NSA, por sus siglas en inglés).

3.1.2. Secure Hash Algorithm (SHA)

El algoritmo SHA-1 fue desarrollado por la NSA, para ser incluido en el estándar DSS (Digital Signature Standard). Produce tramas de 160 bits, a partir de bloques de 512 bits del mensaje original.

El algoritmo es similar al MD5, con la diferencia de que usa la ordenación *big endian*, la cual describe el orden en el cual una secuencia de octetos se almacena en memoria. En una ordenación *big endian*, el valor más significativo de la secuencia se almacena en la dirección más baja (es decir, primero). Se inicializa de igual manera, es decir, añadiendo al final del mensaje un bit en 1 seguido de tantos bits en 0 como sea

necesario hasta completar 448 bits en el último bloque, para luego yuxtaponer la longitud en bits del propio mensaje –en este caso, el primer byte de la secuencia será el más significativo–. A diferencia de MD5, SHA-1 emplea cinco registros de 32 bits en lugar de cuatro:

$$\begin{aligned} A &= 0x67452301 \\ B &= 0xEFCDAB89 \\ C &= 0x98BADCFE \\ D &= 0x10325476 \\ E &= 0xC3D2E1F0 \end{aligned}$$

Los registros anteriores son copiados a cinco variables diferentes $a = A, b = B, c = C, d = D\, y\, e = E$. El ciclo principal tiene cuatro rondas con 20 operaciones cada una. Cada operación realiza una función no lineal sobre tres de las cinco variables *a, b, c, d* y *e*. Al resultado se le suma el valor obtenido después de realizar un desplazamiento circular de cinco bits a una de las otras variables y posteriormente se suma el sub-bloque M_i del mensaje más una constante K_i. La figura 3.5 muestra una operación de SHA, como se puede observar, la actualización de valores de las variables *a, b, c, d* y *e* dependen de ciertas operaciones realizadas sobre los valores actuales de las cinco variables.

La descripción de las funciones no lineales usadas por SHA en cada ronda es la siguiente:

$$\begin{aligned} f_t(X,Y,Z) &= (X \wedge Y) \vee ((\neg X) \wedge Z) \text{ para } t = 0 \text{ hasta } t = 19 \\ f_t(X,Y,Z) &= X \oplus Y \oplus Z \text{ para } t = 20 \text{ hasta } t = 39 \\ f_t(X,Y,Z) &= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \text{ para } t = 40 \text{ hasta } t = 59 \\ f_t(X,Y,Z) &= X \oplus Y \oplus Z \text{ para } t = 60 \text{ hasta } t = 79 \end{aligned}$$

Además se emplean cuatro constantes, una para cada ronda:

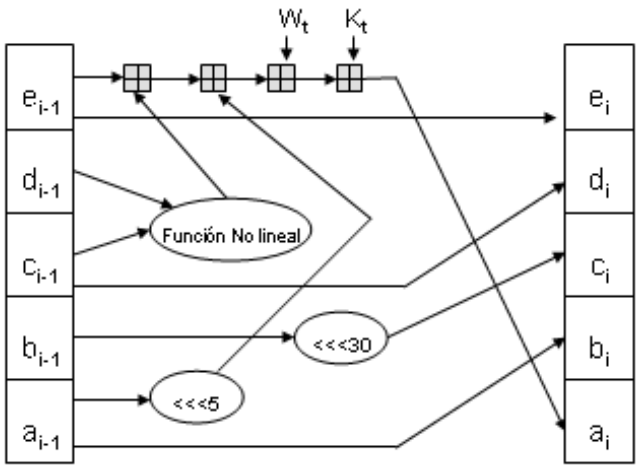

Figura 3.5: Una operación de SHA

$$
\begin{aligned}
K_0 &= 0x5A827999 \\
K_1 &= 0x6ED9EBA1 \\
K_2 &= 0x8F1BBCDC \\
K_3 &= 0xCA62C1D6
\end{aligned}
$$

El bloque *i-ésimo* del mensaje original, M_i, se divide en 16 partes de 32 bits m_0 a m_{15} y se convierte en 80 bloques de 32 bits W_0 a W_{79} usando el siguiente algoritmo:

$W_t = m_t$ para $t = 0 \ldots 15$

$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) / 1$ para $t = 16 \ldots 79$

Todos los m_i obtenidos se interpretan como enteros en las operaciones del algoritmo empleando la ordenación big endian. El ciclo principal del algoritmo es entonces el siguiente:

Para $t = 0$ hasta 79

$T_{mp} = (a <<< 5) + f_t(b, c, d) + e + w_t + K_i$

$e = d$

$d = c$

$c = b/30$

$b = a$

$a = T_{mp}$

Después los valores de *a, b, c, d y e* se suman a los registros *A, B, C, D y E,* respectivamente, y el algoritmo continúa con el siguiente bloque de datos. La salida final de SHA-1 es la concatenación de los registros *A, B, C, D y E*.

Al momento de escribir esta tesis un grupo de criptólogos chinos (Xiaoyun Wang y Hongbo Yu de la Universidad Shandong y Yiqun Lisa Yin de la Universidad de Princeton) rompió el algoritmo SHA-1 basándose en una búsqueda de colisiones usando el ataque del cumpleaños, para mayor información sobre este ataque ver [2], el cual permite obtener dos hash iguales para dos mensajes diferentes con un esfuerzo menor que el de fuerza bruta. Es decir, si un algoritmo hash que genera digestos de 160 bits permite obtener una colisión con un esfuerzo computacional equivalente a 2^{80} operaciones hash, ellos encontraron dicha colisión en SHA-1 con un esfuerzo equivalente a probar 2^{69} veces [34].

3.1.3. Aplicaciones de las funciones hash

Las aplicaciones de las funciones hash con las siguientes:

1. Firmas Digitales: Realizar operaciones de firmas digitales sobre mensajes grandes puede consumir mucho tiempo. En vez de eso, al mensaje se le aplica la función hash y el algoritmo de firma digital se aplica al valor hash obtenido. Con la firma se obtiene autenticación e integridad de los datos.

2. Derivación de llaves. Calcular secuencias para crear nuevas llaves con base en otra llave. Dentro de este punto está el convertir una contraseña a una llave más extensa.

3. Generación de números pseudoaleatorios. Las funciones hash se pueden usar para generar secuencias de números pseudoaleatorios. En la figura 3.6 se muestra un ejemplo de como crear una secuencia con base a un semilla.

3.2. Algoritmos para firma/verificación digital

Como ya se mencionó en el capítulo 1, la firma digital es un mecanismo con el cual se puede ligar una identidad a una pieza de información (ver figura 1.8). La firma digital es análoga a la firma autógrafa en algunos aspectos, tales como:

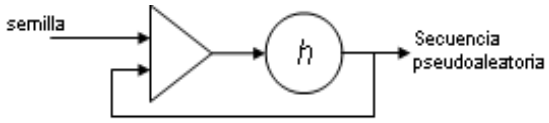

Figura 3.6: Creación de números pseudoaleatorios

- Debe verificar al autor, la fecha y la hora en que se creó la firma.
- Debe autentificar el contenido de la información que va a ser firmada.
- Debe ser comprobable por terceras personas para resolver posibles conflictos.

Sin embargo, la firma digital debe cumplir, además, con las siguientes características [15]:

- La firma debe ser una configuración de bits que depende del mensaje que es firmado.
- La firma debe utilizar cierta información del emisor que sea única, para prevenir la falsificación y la negación.
- Debe ser relativamente fácil producir la firma digital.
- Debe ser relativamente fácil reconocer y verificar la firma digital.
- Debe ser computacionalmente difícil crear un nuevo mensaje para una firma digital existente u obtener una firma digital fraudulenta para un mensaje dado.
- Debe ser práctico conservar una copia de la firma almacenada en algún dispositivo.

Usando los esquemas de criptografía de llave pública se pueden realizar firmas digitales, a continuación se presentan los tres algoritmos más usados hoy en día para llevar a cabo la firma y verificación digital.

3.2.1. PKCS

El procedimiento para firmar digitalmente un documento usando RSA se basa en sus operaciones de cifrado y descifrado. Dichas operaciones están especificadas en el estándar Public Key Cryptography Standard (PKCS #1) [17], el procedimiento para llevar a cabo una firma digital usando RSA se especifíca en PKCS #6 [18].

Para que **A** firme un mensaje lo primero que tiene que hacer es generar su par de llaves en caso de no contar con ellas. Recordando, el procedimiento para generar las llaves es el siguiente:

1. Generar dos números primos grandes p, q, y cualcula $n = pq$.
2. Elegir e_A tal que $1 < e_A < \Phi(n)$ con $mcd(e_A, \Phi(n)) = 1$. Donde $\Phi(n) = (p-1)(q-1)$.
3. Calcular d_A tal que $e_A d_A \equiv 1 (mod\ \Phi(n))$
4. **A** publica (e_A, n) y mantiene privado a d_A, p y q.

Una vez que A tiene su par de llaves, lo siguiente que hace es:

1. **A** pasa el mensaje M una función hash para obtener el digesto $h(M)$.
2. $h(M)$ y el identificador del algoritmo hash utilizado se combinan en un valor ASN.1, después los datos se codifican usando las Reglas Básicas de Codificación (BER, por sus siglas en inglés, definido en X.209).
3. La cadena resultante de la codificación se divide en octetos.
4. Cada octeto se convierte a entero y los valores se concatenan para formar la representación entera de la cadena, m.
5. **A** calcula:
$$s = m^{d_A} \bmod n$$
6. Finalmente **A** convierte s en una cadena de octetos S.
7. **A** envía $(M,\ S)$

Para verificar la firma **B** hace lo siguiente:

1. Rechaza S si su longitud no es múltiplo de 8.
2. **B** convierte la cadena S en un entero s, como en el paso 4 del proceso de firmado.

3. **B** debe calcular:

$$m = s^{e_A} \bmod n$$

4. m se convierte a una cadena de octetos.

5. **B** analiza la cadena para verificar que sea correcta [18].

6. **B** decodifica la cadena para obtener el digesto $h(M)$ y el algoritmo hash utilizado. Rechazar si el algoritmo no es MD2, MD5 o SHA.

7. **B** pasa el mensaje M por la función hash utilizada por **A** para obtener el digesto $h\prime(M)$.

8. Finalmente, **B** compara $h(M)$ con $h\prime(M)$. Si son iguales, la firma es verificada.

3.2.2. DSA

El algoritmo de firma digital, DSA por sus siglas en inglés fue propuesto en 1991 por el Instituto Nacional de Estándares y Tecnología de Estados Unidos. Pero fue hasta 1994 que se adoptó como estándar. DSA es un esquema de firma digital con huella digital o digesto, es decir, sólo se firma el mensaje digerido producto de hacer pasar el mensaje original por una función hash. Como en el caso de la criptografía de llave pública, DSA necesita un par de llaves las cuales se generan de la siguiente manera:

Cada usuario selecciona una llave privada x y genera una llave pública y, donde la llave privada es un número aleatorio o pseudoaleatorio tal que $0 < x < q$ y la llave pública se obtiene como:

$$y = g^x \bmod p$$

Los parámetros que usa el algoritmo son los siguientes:

- Un número primo p donde $2^{L-1} < p < 2^L$ para $512 \leqslant L \leqslant 1024$ y L es un múltiplo de 64.
- Un factor primo de $(p-1)$ denominado q con un longitud de 160 bits.
- $g = h^{(p-1)/q} mod\, p$, donde h es cualquier número entre el rango $1 < h < p-1$, tal que $h^{(p-1)/q} mod\, p > 1$

Los parámetros anteriores son publicados por **A**, para que **B** los conozca.

Para firmar un mensaje m, el procedimiento es el siguiente:

1. **A** genera un número aleatorio k tal que $0 < k < q$.

2. **A** genera:

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ s &= (k^{-1}(H(m) + x_A r)) \bmod q \end{aligned}$$

Donde $H(m)$ es una función hash; el estándar especifica que el algoritmo para la función hash debe ser SHA.

La firma de **A** para el mensaje m es la dupla (r, s), la cual es enviada a la entidad **B** concatenada con el mensaje original *m*.

Para verificar la firma digital, **B** hace lo siguiente:

1. $w = s^{-1} \bmod q$

2. $u_1 = (H(m) * w) \bmod q$

3. $u_2 = (rw) \bmod q$

4. $v = ((g^{u_1} * y_A^{u_2}) \bmod p) \bmod q$

5. Si $v = r$, entonces la firma digital es verificada

3.2.3. ECDSA

El algoritmo de firma digital con curvas elípticas, ECDSA por sus siglas en inglés, es la versión para curvas elípticas del estándar DSA. Fue aceptado en 1999 como un estándar ANSI, y fue adoptado en 2000 por la IEEE y el NIST como estándar. A diferencia de los problemas ordinarios de logaritmo discreto y problemas de factorización, no se conoce algún algoritmo de tiempo subexponencial para el problema de logaritmo discreto con curvas elípticas.

Para firmar un mensaje *m* con ECDSA es necesario que **A** con parámetros de dominio *D = (q, FR, a, b, G, n, h)* y un par de llaves asociadas *(d, Q)* haga lo siguiente:

1. Seleccionar un entero aleatorio k, tal que $1 \leqslant k \leqslant n - 1$

2. Calcular $kG = (x_1, y_1)$ y convertir x_1 a un entero $\overline{x}_1$

3. Calcular $r = x_1$ mod *n*. Si $r = 0$ entonces regresar al paso 1.
4. Calcular k^{-1}mod *n*
5. Calcular la función *SHA-1(m)* y convertir la cadena resultante de bits en un entero e.
6. Calcular $s = k^{-1}(e + dr) \bmod n$. Si $s = 0$ regresar a paso 1.

La firma digital de **A** para un mensaje *m* es la dupla (r, s).

Para realizar la verificación de la firma es necesario que **B** obtenga los parámetros de dominio de la entidad **A**, *D = (q, FR, a, b, G, n, h)* y la llave pública asociada *Q*. Con estos elementos **B** hace lo siguiente:

1. Verificar que r y s sean enteros dentro del intervalo $[1, n - 1]$.
2. Calcular la función hash *SHA-1(m)* y convertir la cadena de bits resultante a un entero e.
3. Calcular $w = s^{-1} \bmod n$.
4. Calcular $u_1 = ew \bmod n$ y $u_2 = rw \bmod n$.
5. Calcular $X = u_1G + u_2Q$.
6. Si $x = O$ (punto en el infinito) entonces rechazar la firma. En otro caso, convertir la coordenada *x*, es decir, x_1 de *X* en un entero $\overline{x}_1$, y calcular $v = \overline{x}_1 \bmod n$.
7. Si $v = r$, la firma es verificada.

De manera resumida, los parámetros de dominio *D* son los siguientes:

1. q es el tamaño del campo finito, donde el valor de q puede ser un número primo impar p o $q = 2^m$.
2. Una indicación FR (representación del campo) de la representación usada para los elementos de F_q.
3. a y b son elementos de F_q con los cuales se define la ecuación de la curva elíptica E sobre F_q (i.e. $y^2 = x^3 + ax + b$).
4. Dos elementos del campo finito x_G y y_G en F_q que definen el punto $G = (x_G, y_G)$ de orden primo en $E(F_q)$.

5. El orden n del punto G, con $n > 2^{160}$ y $n < 4\sqrt{q}$.

6. El cofactor $h = \#E(Fq)/n$

Para más detalles sobre los parámetros de dominio consultar [16].

Capítulo 4

Autenticación y Protocolos de Seguridad

Para proveer los servicios de autenticación y no repudio se puede hacer uso de las firmas digitales. Desafortunadamente el esquema de firma digital no está extento de ataques, por lo que es necesario que una tercera entidad avale la identidad de los participantes en una comunicación. Para ello existen los certificados digitales, en los cuales una entidad de confianza (Autoridad Certificadora) certifica la identidad de una entidad.

También existen protocolos cuyo objetivo es establecer sesiones seguras y, en algunos, casos crear secretos compartidos (llaves) para establecer un intercambio seguro de información. Dichos protocolos pueden establecer sesiones anónimas, es decir, sin necesidad de autenticar a las partes o sesiones con autenticación mutua, donde las entidades involucradas deben identificarse de alguna forma, por ejemplo, vía certificados digitales

En este capítulo se muestra un panorama sobre la autenticación y los protocolos de seguridad. En la primera sección se describen algunos ataques que afectan a la autenticación. En la sección 4.2 se presentan los certificados digitales y se muestran las características del certificado X.509. En la sección 4.3 se habla de los protocolos de seguridad y la descripción de los protocolos Diffie-Hellman, Autenticación por Retos y TLS/WTLS. Finalmente en la sección 4.4 se muestra un modelo analítico propuesto en este tema de tesis para obtener el tiempo de duración estimado del protocolo de negociación de WTLS.

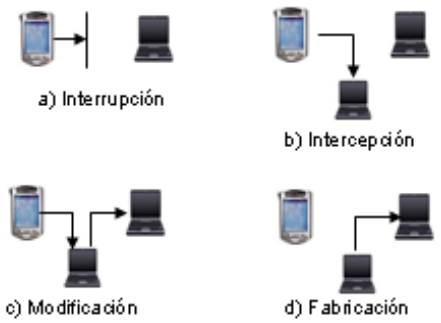

Figura 4.1: Ataques activos a la información y a la autenticación

4.1. Ataques a la autenticación.

Los ataques se pueden clasificar de la siguiente manera:

- **Pasivos**: Estos ataques consisten en escuchar el tráfico que circula por una red, con la intención de obtener cierta información.
- **Activos**: Son aquellos que no solo interceptan la información sino que también manipulan los datos que circulan por la red. La figura 4.1 muestra los ataque que a continuación se describen.

Interrupción: Se presenta cuando se destruye una pieza de software o hardware. Es un ataque a la disponibilidad y la solución es tener un backup o una ruta alterna para la transmisión de la información.

Intercepción: Es un ataque directo a la confidencialidad. Se evita utilizando esquemas de cifrado.

Modificación: Ataque directo a la Integridad. Se evita haciendo uso de Funciones Hash.

Fabricación: Alguien ajeno a la red se hace pasar como miembro y viola el servicio de autenticación. Lógicamente es un ataque directo a la Autenticidad.

A pesar de las ventajas del esquema de firma digital, desafortunadamente, existe una ataque conocido como "intruso de enmedio.º "man in the middle", que puede burlar el esquema de llave pública sin necesidad de romperlo. En la figura 4.2 se puede observar esquematicamente el ataque del intruso de enmedio. **A** desea tener una comunicación con **B**. Tanto **A** piensa que está comunicandose con **B**, como **B** piensa que está en contacto con **A**, pero la realidad es que **E** se hace pasar por **A** y por **B**. **E** en ningún momento rompió el

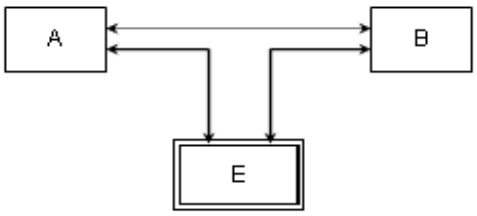

Figura 4.2: Ataque del intruso de enmedio

esquema de llave pública, sólo dió su llave pública tanto a ambas entidades y les hizo creer que esa llave era de la contraparte con la cual estaban en contacto. Es importante autentificar tanto la identidad de la entidad con la cual se establece el contacto y, a su vez, certificar de alguna manera que la llave pública sí pertenece a dicha entidad.

También existe un ataque denominado replay"el cual consiste en reenvían paquetes de información cifrados, cabe señalar que el oponente no necesita conocer la llave que se utilizó para cifrar. Por ejemplo, si el mensaje original era pagar cierta cantidad de dinero y algún oponente repite este mensaje, el resultado será que la entidad que envío el mensaje original deba pagar el doble de dinero.

Uno de los problemas que enfrenta la criptografía de llave pública es que si existe un servidor de llaves pública el cual es accesible a todo mundo, puede darse el caso de que un atacante **C** tome la llave de una entidad **A** y le asigne su nombre como si ésta fuera suya. Si se considera el escenario donde la entidad **C** planea realizar un fraude y elige a la entidad **A** como su victima entonces primero hace creer a **A** que el banco al cual está afiliada requiere alguna información, a su vez, **C** hace la petición al banco para realizar una transacción monetaria. El banco va autenticar a su cliente por medio de un reto, el cual consiste en cifrar cierta información usando la llave pública del cliente, y a su vez dicha información la va a firmar para autenticarse. La llave que utilizará el banco será la de **A** porque fue renombrada como si fuera de **C**. El atacante al recibir el reto se lo envía a la entidad **A** para que esta se autentique y le pide que regrese la firma de la información para comprobar su identidad, el atacante recibe de **A** la respuesta al reto impuesto por el banco y se lo reenvía al banco. El banco al recibir la respuesta del reto enviado autentica a la entidad y procede a realizar la transacción solicitada. El banco cree que su cliente **C** hizo la transacción, pero en realidad la transacción se hizo sobre la cuenta de **A**. Este ataque se denomina "Identity Misbinding.º ataque a la identidad el cual se muestra en la figura 4.3.

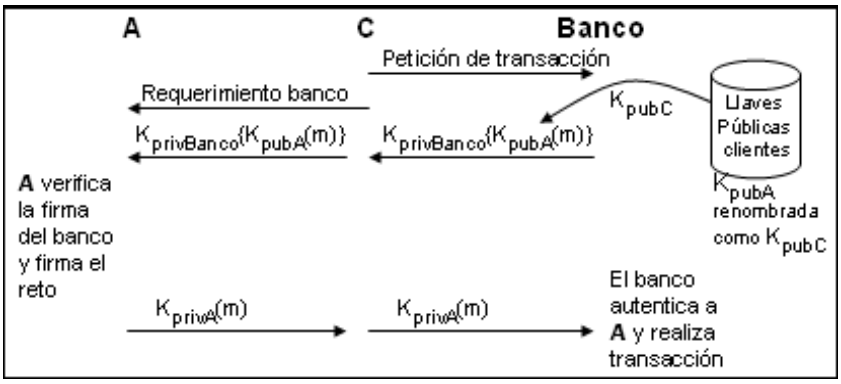

Figura 4.3: Ataque a la identidad

Para evitar los ataques mencionados anteriormente existen protocolos de seguridad, sin embargo, algunos protocolos aún son vulnerables a sufrir ataques y con ello comprometer la información intercambiada o la identidad de las entidades. Uno de los mecanismos que protege de mejor manera la autenticación son los certificados digitales ya que con ellos se certifica a una entidad junto con su llave pública y así evitar que un oponente se haga pasar por otro y perpetrar algún ataque. En las siguientes secciones se muestran las características de los certificados digitales y algunos protocolos de seguridad.

4.2. Autenticación con certificados digitales

Un certificado digital es un documento electrónico en donde se unen la llave pública de una entidad y uno o más atributos referidos a su identidad. El certificado garantiza que la llave pública pertenece a la entidad identificada y que la entidad posee la correspondiente llave privada. Los certificados digitales deben de ser expedidos por una Autoridad Certificadora, AC, ya que si una entidad se certifica a sí misma no hay ninguna garantía de que su identidad sea la que presume, y por lo tanto, no va a ser aceptada por una tercera entidad que no la conozca.

Es importante ser capaz de verificar que una autoridad certificadora ha emitido un certificado y detectar si éste es válido. Para evitar la falsificación de certificados, la entidad certificadora los firma digitalmente después de autentificar la identidad de un sujeto.

Si el certificado es auténtico y se confía en la AC, entonces, también se confía en que el sujeto iden-

tificado en el certificado digital posee la llave pública que se señala en dicho certificado. Así, si un sujeto firma un documento y anexa su certificado digital, cualquiera que conozca la llave pública de la AC podrá autenticar el documento.

Como se puede observar, los certificados digitales son una herramienta muy eficiente para lograr la autenticación entre dos o más entidades, con la premisa de que esas entidades deben de confiar en una tercera entidad, es decir, la AC. Existe un formato para creación de certificados digitales ampliamente conocido, el formato X.509 que se discute a continuación.

4.2.1. Certificados X.509

El formato de los certificados X.509 es un estándar del International Telecommunication Union Telecommunication Standarization Sector (ITU-T) y el International Standards Organization / International Electrotechnical Commission (ISO/IEC) que se publicó en 1988. Actualmente la versión que está en uso es la 3. El formato de la versión 1 fue extendido en 1993 para incluir dos nuevos campos que permiten soportar el control de acceso a directorios. Después de emplear el X.509 v2 para intentar desarrollar un estándar de correo electrónico seguro, el formato fue revisado para permitir la extensión con campos adicionales, dando lugar al X.509 v3, publicado en 1996. En figura 4.4 se muestra el esquema de un certificado digital X.509.v3.

Los elementos principales de un certificado X.509 v3 son:

- **Versión**. Contiene el número de versión del certificado codificado. Los valores aceptables son 1, 2 y 3.
- **Número de serie del certificado**. Es un entero asignado por la autoridad certificadora. Cada certificado emitido por una AC debe tener un número de serie único.
- **Identificador del algoritmo de firmado**. Identifica el algoritmo empleado para firmar el certificado (ECC, RSA o DSA).
- **Nombre del emisor**. Identifica a la AC que ha firmado y emitido el certificado.
- **Período de validez**. Indica el período de tiempo durante el cual el certificado es válido y la AC está obligada a mantener información sobre el estado del mismo. El campo incluye la fecha en la que el certificado fue expedido y la fecha de revocación.

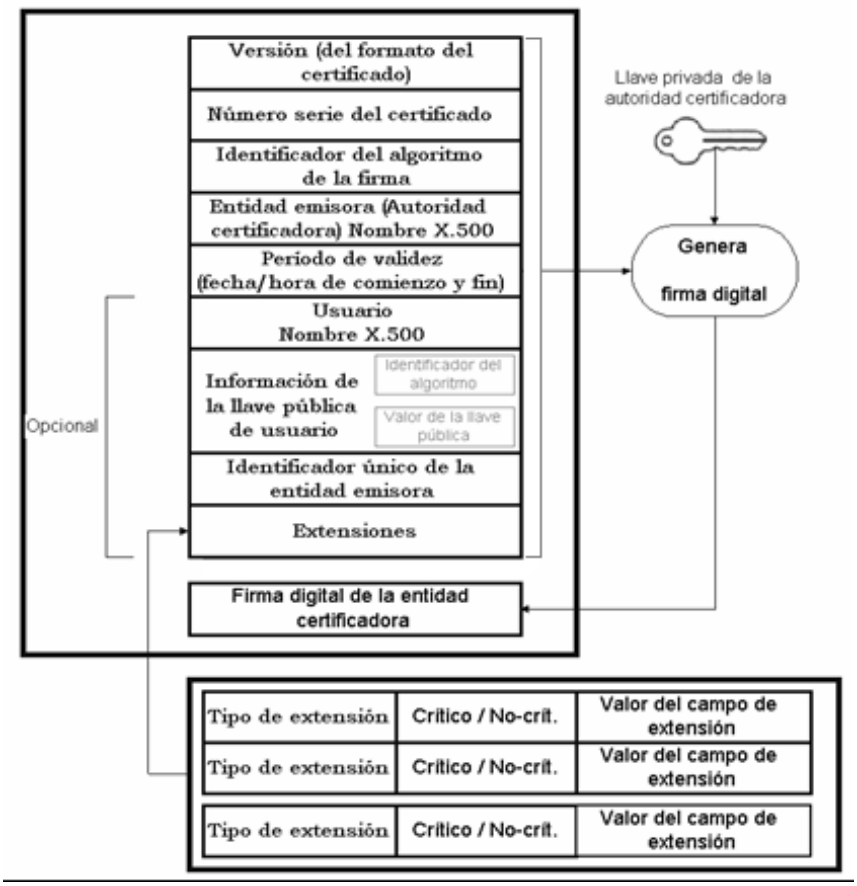

Figura 4.4: Certificado digital X.509

- **Nombre del sujeto**. Este campo identifica la identidad cuya llave pública está certificada en el campo siguiente. El nombre debe ser único para cada entidad certificada por una AC dada, aunque puede emitir más de un certificado con el mismo nombre si es para la misma entidad.
- **Información de llave pública del sujeto**. Contiene la llave pública, sus parámetros y el identificador del algoritmo con el que se emplea la llave.
- **Identificador único del emisor**. Este es un campo opcional que permite reutilizar nombres de emisor.
- **Identificador único del sujeto**. Este es un campo opcional que permite reutilizar nombres de sujeto.

Las extensiones del X.509 v3 proporcionan una manera de asociar información adicional a sujetos, llaves públicas, por mencionar algunos. Un campo de extensión tiene tres partes:

- **Tipo de extensión**. Es un identificador de objeto que proporciona la semántica y el tipo de información (cadena de texto, fecha u otra estructura de datos) para un valor de extensión.
- **Valor de la extensión**. Este subcampo contiene el valor actual del campo.
- **Indicador de importancia**. Es una bandera que indica a una aplicación si es seguro ignorar el campo de extensión si no reconoce el tipo. El indicador proporciona una manera de implementar aplicaciones que trabajan de modo seguro con certificados y evolucionan conforme se van añadiendo nuevas extensiones.
- **Firma digital**. El certificado es firmado por la AC.

El formato de los certificados X.509 se especifica en un sistema de notación llamada Abstract Sintax One (ASN-1). La Notación de Sintaxis Abstracta 1, es un lenguaje para describir los mensajes que se intercambian las aplicaciones en internet, ejemplos de aplicaciones que usan ASN.1 son: redes inteligentes, teléfonos celulares, comercio electrónico, servicios electrónicos seguros, etc.

4.3. Protocolos de seguridad

El objetivo de algunos de los protocolos de seguridad, además de autenticar a las entidades, es crear un secreto compartido denominado llave de sesión K_s, la cual es un número no muy grande (al menos de

80 bits para que sea considerada segura). Dicha llave se usa para cifrar y descifrar utilizando criptografía de llave secreta por un periodo no muy grande. A diferencia de las llaves públicas, una llave de sesión sólo existirá durante el tiempo que dura el intercambio de datos. Para crear una llave de cifrado/descifrado existen algunos mecanismos.

1. Una entidad **A** genera un número aleatorio el cual será la llave de sesión y de alguna manera se lo hace llegar a **B**. Una de las formas para el intercambio de este número es usando criptografía de llave pública. Una aplicación que hace uso de este mecanismo es PGP. Dicha herramienta genera un número aleatorio que se utiliza para cifrar datos, el número es cifrado usando criptografía de llave pública y es enviado conjuntamente con los datos cifrados para que el receptor descifre el número con su llave privada y con él poder descifrar los datos.

2. **A** y **B** hacen uso de un protocolo de negociación. El objetivo de dichos protocolos es establecer ciertos parámetros en común entre ambas entidades para que con base a ellos se genere una llave.

A continuación se muestra la descripción de algunos de los protocolos de seguridad que existen en la actualidad.

4.3.1. Autenticación por retos

Un Protocolo de Negociación para Autenticación por Retos (CHAP, por sus siglas en ingles) es un esquema donde el agente autenticador (típicamente un servidor en una red) envía a un programa cliente un valor aleatorio que es usado una sola vez y un valor de identificación. Tanto el emisor como el receptor comparten un secreto previamente establecido. El valor aleatorio, la identificación y el secreto se concatenan y se hacen pasar por una función hash, usando MD5. El valor obtenido de la función hash se envía al agente autenticador, quien a su vez realizó el mismo procedimiento. Si los valores son iguales se ha autenticado al programa cliente. La figura 4.5 muestra el esquema de este protocolo.

4.3.2. Diffie-Hellman

El protocolo de Diffie-Hellman fue el primer algoritmo de llave pública [2], el cual se dió a conocer en 1976 en [3]. Dicho algoritmo basa su seguridad en la dificultad de calcular el logaritmo discreto en un campo finito. Sin embargo, es fácil calcular la exponenciación en el mismo campo. El propósito del algoritmo es

Figura 4.5: Autenticación por retos

crear una llave secreta entre, al menos, dos entidades. Posteriormente, dicha llave puede usarse para cifrar y descifrar mensajes.

Si las entidades **A** y **B** desean generar una llave, lo primero que deben hacer es ponerse de acuerdo sobre la utilización de un número primo grande n, y crear un generador g, tal que sea un número primitivo mod n. Donde un número es primitivo si cumple con:

Si n es primo, y $g < n$, entonces g es un generador $mod\, n$, si para cada b en el intervalo $[1, p-1]$, existe algún número a donde $g^a \equiv b(mod\, n)$.

Ya que ambas entidades tiene los parámetros públicos n y g hacen lo siguiente:

1. **A** elige un número aleatorio grande x.
2. **A** calcula $X = g^x mod\, n$ y lo envía a **B**.
3. **B** elige un número aleatorio grande y.
4. **B** calcula $Y = g^y mod\, n$ y lo envía a **A**.
5. **A** calcula $k = Y^x mod\, n$
6. **B** calcula $k\prime = X^y mod\, n$

Tanto k como $k\prime$ son igual a $g^{xy} mod\, n$.La figura 4.6 muestra el protocolo Diffie-Hellman.

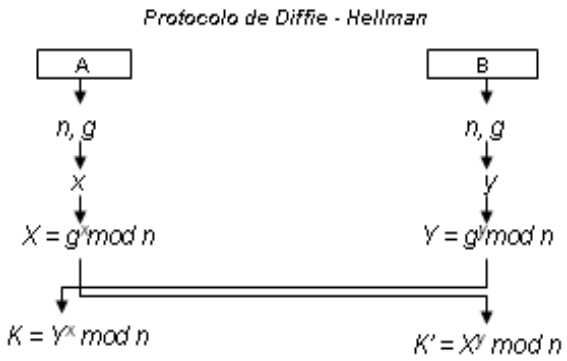

Figura 4.6: Protocolo Diffie-Hellman

4.3.3. PGP

Pretty Good Privacy (PGP), es un proyecto iniciado en 1991 por Phill Zimmerman. La motivación para realizar este proyecto fue la ausencia de herramientas sencillas, potentes y baratas que acercaran la criptografía seria a los usuarios comunes. Poco a poco PGP se convirtió en un estándar de facto para sistemas de seguridad para intercambio de datos y correo seguro.

PGP trabaja con una combinación de criptografía de llave privada y criptografía de llave pública otorgando una gran facilidad al usuario para gestionar sus llaves públicas y privadas. Para trabajar con criptografía de llave pública es necesario poseer las llaves públicas de todos los posibles receptores de mensajes, además es necesario tener la llave privada propia. Para poder gestionar tanto las llaves públicas como privadas PGP utiliza anillos de llaves o llaveros, que no son más que archivos que PGP proporciona al usuario para poder guardar las llaves públicas (pubring) y las llaves privadas (secring). PGP genera llaves de sesión para poder utilizar la criptografía de llave secreta. Las llaves de sesión se generan aleatoriamente cada vez que un usuario quiere cifrar un archivo.

Cifrado.

De manera general el esquema de cifrado que utiliza PGP se puede observar en la figura 4.7.Al tener la llave de sesión recién generada, el texto en claro se hace pasar por un algoritmo de cifrado simétrico, en este caso IDEA, y posteriormente, la llave de sesión se cifra usando RSA utilizando la llave pública del destinatario. Cuando dicha llave de sesión se ha cifrado se concatena con el texto cifrado y se envía vía correo electrónico al destinatario.

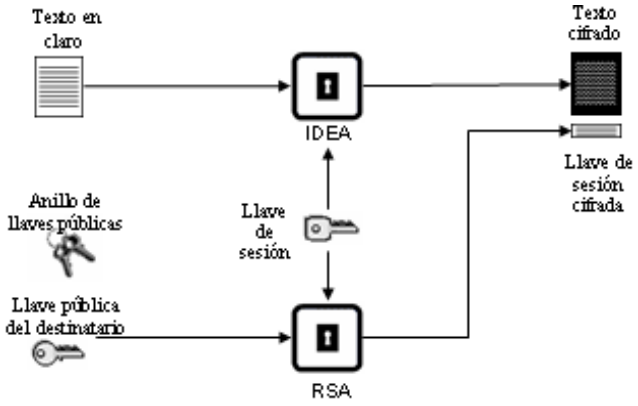

Figura 4.7: Esquema de cifrado en PGP

Descifrado.

Para descifrar, PGP pide al usuario una contraseña para que ésta se haga pasar por una función hash, MD5, y así crear una llave para poder descifrar la llave privada del usuario. Ya que se ha descifrado la llave privada usando un algoritmo de cifrado simétrico, IDEA, se procede a descifrar la llave de sesión usando RSA. Con la llave de sesión descifrada, el siguiente paso es descifrar el mensaje usando el mismo algoritmo de cifrado usado por el emisor, es decir, IDEA. El esquema de descifrado se muestra en la figura 4.8.

Firmado digital de archivos.

Lo primero que se obtiene es la huella digital o digesto del documento que se desea firmar, es decir, el documento se hace pasar por una función hash , en este caso MD5, aunque versiones actuales de PGP ya usan SHA-1 y DSA. Dicho digesto se cifra usando RSA utilizando la llave privada del usuario. Finalmente, la firma digital se concatena con el documento y se envía al destinatario. La figura 4.9 muestra el esquema de firma digital usado en PGP.

Verificación digital.

Para verificar un archivo, el usuario al recibir el archivo firmado debe descifrar el digesto usando RSA con su llave pública. Asimismo, se debe obtener el digesto asociado al documento que se recibio usando el mismo algoritmo hash. Una vez que se tienen los dos digestos, se procede a compararlos, si son iguales, la firma es verificada, en otro caso, la firma no es válida. El esquema de verificación digital usado por PGP se muestra en la figura .

Desventajas de PGP.

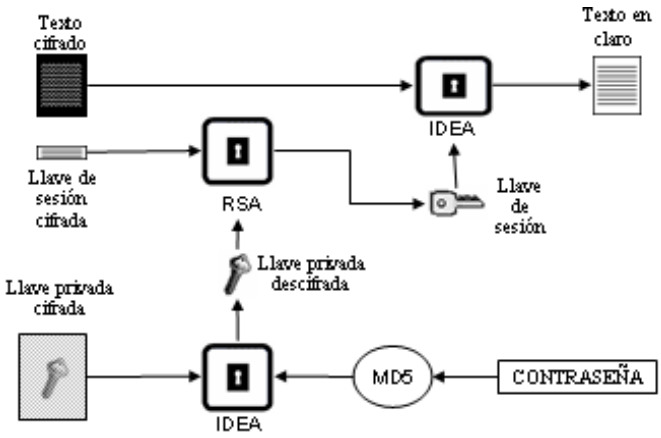

Figura 4.8: Esquema de Descifrado en PGP

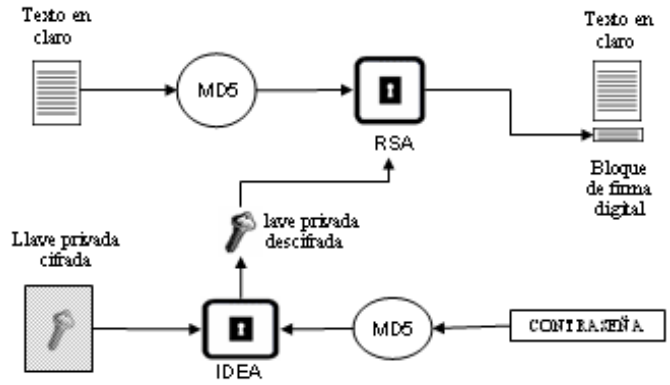

Figura 4.9: Esquema de firmado digital de documentos en PGP

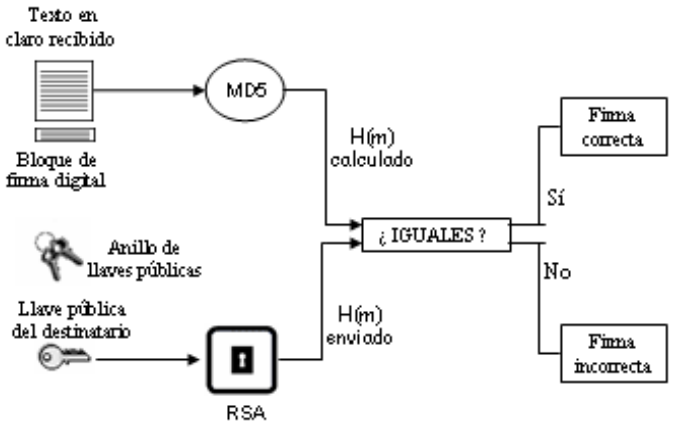

Figura 4.10: Esquema de verificación digital en PGP

- La llave de sesión viaja por el canal inseguro. Aunque dicha llave se cifre con RSA, puede darse el caso de un ataque y recuperarla.
- Las versiones existentes de PGP actualmente no contemplan el criptosistema de curvas elípticas.
- El intercambio de datos es sin conexión, lo que ocasiona que si dos usuarios desean intercambiar varios documentos entre sí, necesitan crear una llave de sesión por cada envío de información.
- Hasta el momento PGP Mobile, es decir, la versión para PDA's, no contempla la utilización de ECC como uno de sus algoritmos de llave pública.

4.3.4. TLS/WTLS

El objetivo principal del protocolo WTLS es proporcionar a las aplicaciones Privacidad, Integridad y el servicio de Autenticación. Esos servicios son alcanzados por la arquitectura de WTLS que fue inspirada en el protocolo TLS, que es el estándar de facto para transacciones seguras en Internet.

Para establecer una conexión segura, WTLS realiza un esquema de negociación conocido como **protocolo de negociación**. En el protocolo de negociación de WTLS, el cliente y el servidor, convienen mutuamente los algoritmos criptográficos que se utilizarán durante la conexión, junto con la autentificación opcional del servidor (o la autentificación de ambos, del cliente y del servidor), vía certificados digitales. Una vez que

esa autentificación mutua se ha logrado con éxito, ambas partes proceden a crear con seguridad un secreto compartido conocido como secreto maestro o llave de sesión. Después de eso, las dos partes pueden iniciar el intercambio de información valiosa de forma segura.

Actualmente, WTLS se apoya solamente en dos criptosistemas de llave pública: RSA y ECC. Para los algoritmos de cifrado simétrico se contemplan DES, TDES, AES, RC4 y A5. Para el caso de los algoritmos hash se tiene MD5 y SHA-1.

Dependiendo de las opciones elegidas en el protocolo de negociación se tienen tres clases de implementaciones de WTLS definidas en la especificación de WAP-261-WTLS-20010406-a Versión 6-Abril-2001 [19], estas son:

- WTLS Clase 1: Sólo brinda privacidad e integridad de datos mediante un intercambio de llaves anónimo sin autenticación.
- WTLS Clase 2: Se provee privacidad e integridad de datos además de autenticación WAP a nivel del servidor. Aquí, la autenticación del servidor se basa en Certificados digitales. La llave del servidor puede ser anónima o autenticada, la llave del cliente es anónima.
- WTLS Clase 3: Se provee privacidad e integridad de datos además de autenticación tanto del servidor como del cliente. La autenticación del servidor como la del cliente se basa en certificados digitales. La llave del servidor y del cliente puede ser anónima o autenticada.

La figura 4.11 muestra el flujo del protocolo de negociación para la clase 3 (es decir, autenticación mutua de cliente y servidor). El significado de los mensajes intercambiados es como sigue.

Primero, un cliente inicia el protocolo de negociación enviando una petición de conexión. Después que el servidor acepta la petición, el cliente procede a enviar un mensaje denominado HolaDelCliente. Este mensaje incluye una sugerencia de la versión a utilizar del protocolo, una lista de las opciones aceptables para cifrar y los métodos de compresión, un valor aleatorio del cliente usado para prevenir ataques y para generar la llave de sesión, y otros parámetros específicos de extensión. Después de enviar ese mensaje, el cliente espera el mensaje correspondiente HolaDelServidor. En el mensaje del servidor se elige la opción de cifrado que se utilizará, una identificación de la versión del protocolo y de la sesión, e incluye un valor aleatorio definido por el servidor más algunos parámetros específicos de extensión adicionales.

El servidor entonces envía su certificado digital junto con un mensaje de PeticiónDeCertificado que enumera los tipos aceptables de los certificados para que el cliente se autentique. El cliente entonces procede a verificar el certificado del servidor y entonces ejecuta el algoritmo de intercambio de llave negociado para

calcular el secreto pre-maestro. Después, el cliente calcula el secreto principal (llave de sesión) y envía al servidor cuatro mensajes: a) el certificado del cliente; b) El mensaje de intercambio de la llave del cliente con la información adicional requerida para establecer la llave de sesión; c) El mensaje de verificación del certificado que incluye la firma del cliente de los mensajes intercambiados hasta este punto y; d) El mensaje de especificación de cambio de cifrado, indicando la intención del cliente de cambiar los parámetros negociados. Finalmente, el cliente envía un mensaje cifrado de terminación.

De forma recíproca, el servidor envía un mensaje de especificación de cambio de cifrado y un mensaje de terminación que incluye un resumen de todos los mensajes intercambiados durante el protocolo. Después de esto, cliente y servidor utilizan la aplicación para intercambiar datos sobre la sesión que se ha establecido con los servicios de confidencialidad y de autentificación.

Los parámeros criptográficos que propone el cliente y que se indican en el mensaje HolaCliente son los siguientes:

- Versión de WTLS: Sugerencia de la versión a utilizar del protocolo.
- Número aleatorio de Seguridad: Es un valor aleario que se genera cada que se inicia una petición de sesión. Sirve además para generar la llave de sesión.
- Identificador de la sesión: Identifica la sesión segura. Ésta es única por servidor.
- Algoritmo para Intercambio de llave: Especifican los parámetros del sistema de generación de llave de sesión y el tipo de autentificación a utilizarse. El estándar contempla los siguientes escenarios.

{*NULLKES, ECDH_anon, RSA_anon, ECDH_oneway, RSA_oneway, ECDH_twoway, RSA_twoway,* **ECDH_twowayHYBRID**}

donde:

NULLKES. No existira intercambio de llaves.

ECDH_anon. El intercambio de llaves será sin autenticación de ninguna de las partes y la llave de sesión se genera con ECDH.

RSA_anon. No hay autenticación de las partes pero el servidor envía su llave pública para que el cliente genere un secreto y lo cifre con ella. Posteriormente dicho secreto se envía al servidor para que éste genere así la llave de sesión.

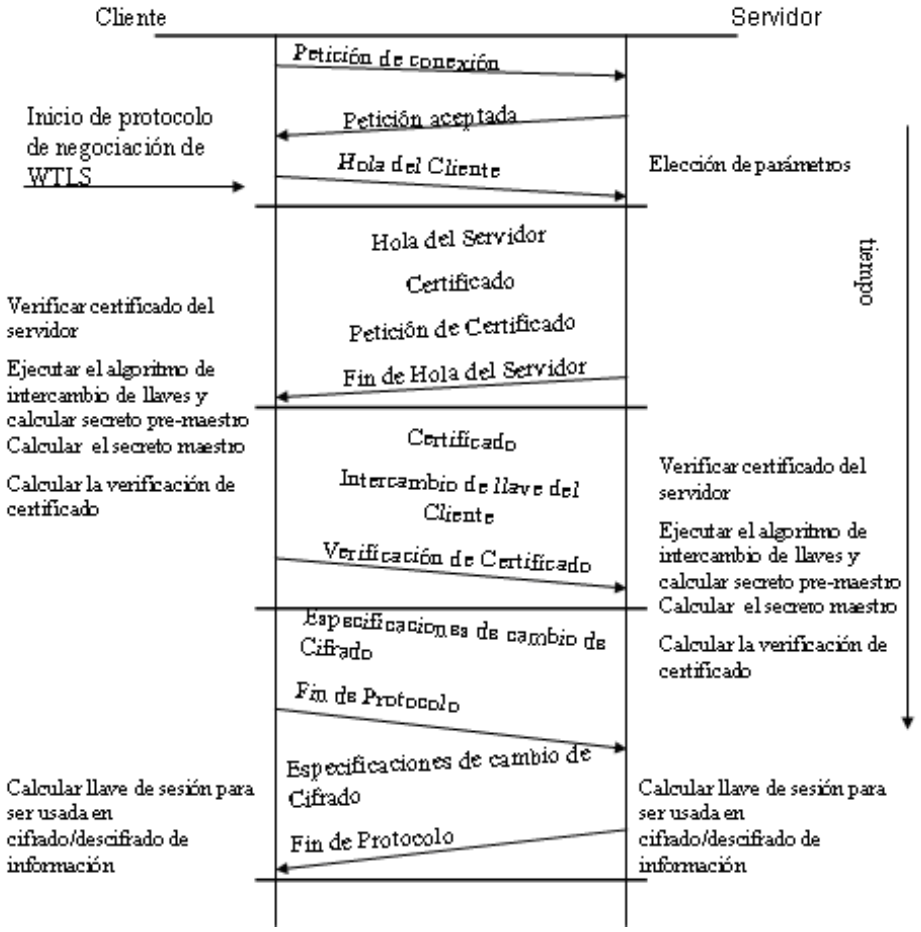

Figura 4.11: Protocolo de Negociación de WTLS

ECDH_oneway. Existe autenticación vía certificado digital firmado con ECDSA por parte del servidor solamente. La creación de la llave de sesión es con base a ECDH.

RSA_oneway. Existe autenticación usando certificados RSA solo del servidor. La llave de sesión se genera de la misma forma que en el caso de RSA_anon, solo que la llave pública del servidor se extrae del certificado digital.

ECDH_twoway. La autenticación es mutua, tanto cliente como servidor envían un certificado firmado usando ECDSA. La generación de la llave de sesión es usando ECDH.

RSA_twoway. Ambas entidades se autentican usando certificados RSA. La generación de la llave de sesión es similar a los casos anteriores de RSA.

ECDH_twowayHIBRID. Esta modalidad es una contribución de este trabajo de tesis al protocolo de negociación de WTLS. En este caso existe autenticación mutua de las entidades usando un certificado digital que contiene la llave pública de la entidad la cual es generadas con ECC. El certificado digital es firmado usando RSA. Para generar la llave de sesión se usa ECDH.

- Algoritmo hash: Especifíca la función hash que se va a utilizar a lo largo de la sesión. En este caso solo se uso SHA-1

- Algoritmo simétrico: En este parámetro el cliente especifíca que algoritmo de cifrado simétrico ha elegido. Las opciones con las cuales cuenta el cliente son:

  { A5, RC4, DES, Triple-DES, AES-128, AES-256}

Para mayor información sobre WTLS consultar el Apéndice B.

4.4. Modelo analítico.

Dado que es muy importante la relación de seguridad contra tiempo en los dispositivos móviles, es necesario considerar las mejores alternativas criptográficas para realizar sistemas de seguridad eficientes. En este trabajo se propone un modelo para el análisis del protocolo de negociación en la clase de implementación 3 que considera el tiempo de ejecución de las operaciones criptográficas realizadas en el cliente y en el servidor durante dicho protocolo.

Símbolo	Descripción
T_{ECC_firma}	Firma digital con ECC
T_{ECC_ver}	Verificación digital con ECC
T_{ECDH}	Diffie-Hellman con Curvas Elípticas
$T_{RSA\_cifrado}$	Cifrado con RSA
$T_{RSA\_descifrado}$	Descifrado con RSA
T_{RSA_firma}	Firma digital con RSA
T_{RSA_ver}	Verificación digital con RSA
T_{gen_skey}	Generación de la llave de sesión
T_{C_MA}	Protocolo de Negociación del Cliente
T_{S_MA}	Protocolo de Negociación del Servidor
T_{COM}	Latencia de la comunicación
$T_{conexión}$	Tiempo de espera para obtener conexión
T_{TCOM}	Estado latente total de la comunicación
T_{HSPROT}	Tiempo total Protocolo de Negociación

Tabla 4.1. Definición de Símbolos

La notación usada en el modelo se resume en la tabla 4.1. En la puesta en práctica, el tiempo total del protocolo de negociación es dado por la adición del tiempo de ejecución total del cliente y del servidor más los gastos indirectos introducidos por el tiempo del estado latente de la comunicación. Por lo tanto el costo de cómputo del protocolo negociación de WTLS se puede estimar como:

$$T_{HSPROT} = T_{C\_MA} + T_{S\_MA} + T_{TCOM} \qquad (1)$$

En una primera aproximación, el tiempo del estado latente de la comunicación se puede modelar como:

$$T_{TCOM} = T_{conexión} + \sum_{1}^{k} T_{COM} \qquad (2)$$

Donde el número *k* es el número total de mensajes intercambiados durante el protocolo de negociación y $T_{conexión}$ es el tiempo de espera por parte del cliente para que su petición de conexión sea atendida por parte del servidor. Los otros dos componentes de la ecuación (1) se estiman de acuerdo al esquema de la llave pública seleccionado, RSA o ECC. En el resto de esta sección damos una valoración analítica de esos dos componentes.

Protocolo de negociación usando RSA

La figura 4.11, muestra como el cliente realiza dos operaciones de llave pública, una cuando verifica el certificado del servidor y otra cuando cifra el secreto pre-maestro. Realiza además una operación de llave privada para generar el mensaje de verificación del certificado.

El servidor también ejecuta dos operaciones de llave pública. Primero, cuando verifica el certificado del cliente y en segundo lugar cuando verifica la firma del cliente. El servidor ejecuta además una operación de llave privada para descifrar el secreto pre-maestro.

Por lo tanto, las ecuaciones (3) y (4) son la estimación del tiempo de ejecución del cliente y del servidor en términos de las operaciones criptográficas realizadas.

$$T_{C\_MA} = T_{RSA\_ver} + T_{RSA\_cifrado} + T_{RSA\_firma} + T_{gen\_skey} \quad (3)$$

$$T_{S\_MA} = 2 * T_{RSA\_ver} + T_{RSA\_descifrado} + T_{gen\_skey} \quad (4)$$

Protocolo de negociación usando ECC

Cuando se utiliza ECC, el cliente realiza tres operaciones criptográficas. Primero verifica el certificado del servidor, en segundo lugar, se realiza una operación de Elliptic Curve Diffie-Hellman (ECDH) para calcular el secreto pre-maestro y finalmente realiza una firma Elliptic Curve Digital Signature Algorithm (ECDSA) para generar el mensaje de verificación para el servidor.

El servidor necesita realizar una operación ECDH para calcular el secreto pre-maestro y después realiza dos verificaciones ECDSA, una para verificar el certificado del cliente y la otra para verificar el mensaje de la verificación del cliente.

Las ecuaciones siguientes estiman el tiempo de ejecución del cliente y del servidor en términos de las operaciones criptográficas realizadas.

$$T_{C\_MA} = T_{ECC\_ver} + T_{ECDH} + T_{ECC\_firma} + T_{gen\_skey} \quad (5)$$

$$T_{S\_MA} = 2 * T_{ECC\_ver} + T_{ECDH} + T_{gen\_skey} \quad (6)$$

Protocolo de negociación híbrido

Dado que RSA y ECC tienen sus ventajas y desventajas, se propone una combinación de ambos criptosistemas para reducir el costo computacional dentro del cliente. Se puso especial atención al cliente dado que en nuestro caso es un dispositivo móvil ligero con restricciones computacionales.

Un criptosistema se define como más predominante porque se utilizan más operaciones criptográficas de dicho criptosistema. En el esquema híbrido que proponemos ECC va a ser predominante ya que se realizarán 6 operaciones de ECC contra sólo 2 operaciones de RSA a lo largo de todo el protocolo de negociación.

ECC es predominante.

El cliente realiza tres operaciones criptográficas, primero se va a verificar el certificado del servidor usando RSA, firma con ECC el mensaje de verificación del certificado y finalmente realiza la operación ECDH.

El servidor también realiza 3 operaciones criptográficas, va a verificar el certificado del cliente usando RSA, verifica el mensaje de verificación de certificado usando ECC y finalmente ejecuta la operación ECDH.

$$T_{C\_MA} = T_{RSA\_ver} + T_{ECC\_firma} + T_{ECDH} + T_{gen\_skey} \quad (7)$$

$$T_{S\_MA} = T_{RSA\_ver} + T_{ECC\_ver} + T_{ECDH} + T_{gen\_skey} \quad (8)$$

En el caso de la autentificación mutua cliente/servidor las ecuaciones anteriores combinadas con la ecuación (1) nos permiten estimar el tiempo de ejecución total para las operaciones criptográficas.

Con este modelo analítico se pretende estimar el tiempo aproximado de ejecución del protocolo de negociación de WTLS. A continuación se presentan las pruebas realizadas y los tiempos de ejecución registrados para el protocolo de negociación. Dichos tiempos serán comparados con los tiempos estimados calculados con base en el modelo analítico para verificar la efectividad del modelo.

Capítulo 5

Diseño e Implementación del Sistema de Seguridad para Intercambio de Datos en Dispositivos Móviles.

A través de los capítulos anteriores, ha quedado de manifiesto que la escasa seguridad con la que están dotados los sistemas de comunicación basados en tecnología inalámbrica es su principal inconveniente. Los protocolos que se utilizan en estos sistemas no cuentan con la robustez suficiente para asegurar que la información que se transmita no será alterada y/o hurtada por entidades no autorizadas. Asimismo, ha quedado claro que en materia de seguridad la criptografía es una herramienta sumamente útil.

En este trabajo de tesis, lo que se buscó es proveer a la cada vez más popular tecnología inalámbrica con la seguridad que la criptografía ofrece. Para ello, se desarrolló un sistema de software que se implementó en dos dispositivos, uno de ellos un dispositivo móvil ligero (un PDA) y el otro una PC (de escritorio o laptop), los cuales entablan una negociación, haciendo uso de WTLS, para crear un secreto compartido (llave de sesión) y así intercambiar datos en un ambiente inalámbrico de forma segura. Cabe mencionar que dentro de los experimentos realizados, se usaron dos dispositivos móviles como clientes que entablaron una sesión segura con el servidor simultáneamente. Además, se cuenta con un esquema de autenticación basado en certificados digitales los cuales están apegados al estándar X.509.

En este capítulo se presenta la descripción del prototipo desarrollado en este trabajo de tesis. Para tal fin, la sección 5.1 expone el diseño del sistema, es decir, se da una descripción del mismo y se detalla su

arquitectura. En la sección 5.2 se muestran algunos detalles de la implementación.

5.1. Diseño del sistema

En esta sección se describe el diseño del Sistema de Seguridad para Intercambio de Datos en Dispositivos Móviles. Se presenta una descripción del mismo, así como la arquitectura y los diagramas de los módulos que lo constituyen.

5.1.1. Descripción del sistema.

El sistema está basado en la arquitectura cliente-servidor en la cual, algunos de los dispositivos de cómputo que están inmersos en la red ofrecen servicios (servidores) y otros los usan (clientes). Para este caso de estudio, existe un dispositivo cliente (un asistente personal digital, PDA) y un dispositivo servidor (una Laptop o una PC), ambos con acceso a la red inalámbrica. Para establecer una sesión de intercambio de datos es necesario que el sistema resida en los dos dispositivos con la configuración adecuada. Debido a las restricciones de recursos (memoria, procesador, espacio de almacenamiento, etc.), un dispositivo móvil ligero (una IPAQ, en este caso) no puede tomar el rol de servidor porque la demanda de recursos de cómputo es mayor.

La conexión inalámbrica se llevo a cabo creando una red ad-hoc entre el servidor y los clientes. Para ello se establecieron direcciones IP fijas tanto para el servidor como para los clientes. En el caso de los clientes se especificó como puerta de enlace la dirección IP del servidor. La figura 5.1 muestra el esquema de la conexión usada en este proyecto de tesis.

El sistema consta de dos módulos principales: el Módulo de Negociación (MN) y el Módulo para Intercambio de Datos (MID). Dichos módulos varían dependiendo del dispositivo en el que residan, es decir, en el cliente se utilizarán procesos que en el servidor no y viceversa. La figura 5.2 muestra el diagrama general del sistema, dicha figura se repite en este capítulo por su importancia, dado que también se mostro en el capítulo 1. Como se puede observar en dicha figura, el usuario, un cliente, utiliza el sistema para intercambiar datos de forma segura. Para ello, es necesario que elija una serie de parámetros criptográficos, tales como, el tamaño de llave para RSA, nombre de la curva elíptica para ECC, algoritmo para firma digital, algoritmo para intercambio de llave, cifrador simétrico, tipo de implementación del protocolo de negociación, los cuales son utilizados por el Protocolo de Negociación de WTLS para establecer una sesión segura. La

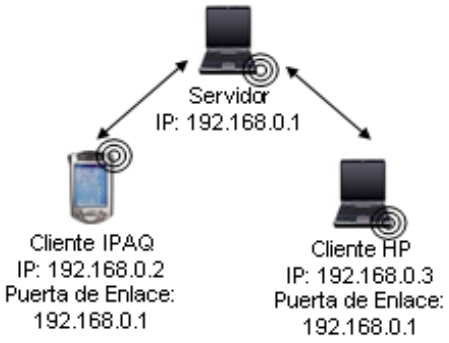

Figura 5.1: Red ad-Hoc

implementación del protocolo de negociación de WTLS se apegó a las especificaciones descritas en [19], dicha implementación reside específicamente en el MN. El objetivo de entablar una negociación es crear un secreto compartido entre cliente y servidor, que se denomina *llave de sesión.* Dicha llave será utilizada por el MID para las operaciones de cifrado y descifrado de datos ofreciendo así el servicio de confidencialidad, mientras que para ofrecer los servicios de integridad y autenticación se utilizan las operaciones de firma y verificación digital. El MID estará activo mientras dure la sesión y durante este tiempo se puede estar intercambiando datos, por ejemplo, archivos de texto, archivos de programas, imágenes, archivos de audio y video, por mencionar algunos. Si la sesión termina por factores ajenos al usuario o por decisión del mismo, será necesario realizar otra negociación para establecer nuevamente una sesión segura.

A diferencia de PGP (ver sección 4.3.5), el sistema que se propone en este tema de tesis no envía la llave de sesión por el canal inseguro, como ya se ha mencionado, dicha llave se crea con base a una negociación la cual cuenta con una autenticación mutua usando certificados para evitar ataques. El crear la llave con base a un protocolo de seguridad conlleva una modificación en la metodología para intercambiar datos de manera confidencial. En la figura 5.3 se puede observar el esquema de cifrado del Sistema de Seguridad para Intercambio de Datos en Dispositivos Móviles. El tiempo para realizar todo el proceso es menor que en PGP dado que sólo se cifra el documento y se envía al destinatario, además de que la transferencia es en línea. Otra ventaja en el cifrado es que se tienen varios cifradores disponibles, la elección del cifrador dependerá completamente de la capacidades del cliente, es decir, si su poder de cómputo es muy restringido, puede elegir un cifrador por flujo de datos y ECC, sino puede elegir un algoritmo más robusto, como por ejemplo

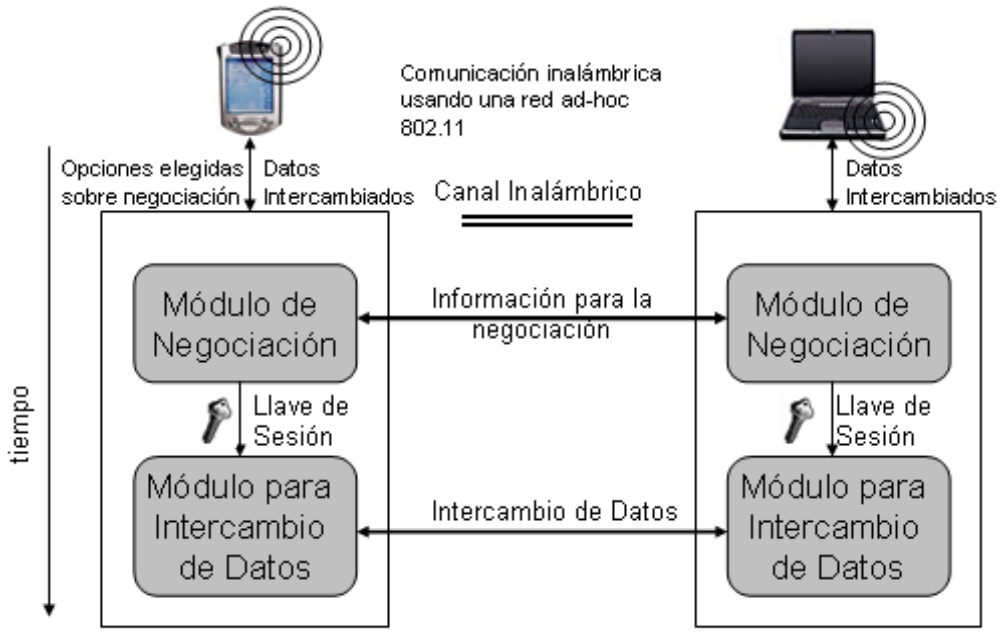

Figura 5.2: Diagrama general del sistema de seguridad para intercambio de datos en dispositivos móviles.

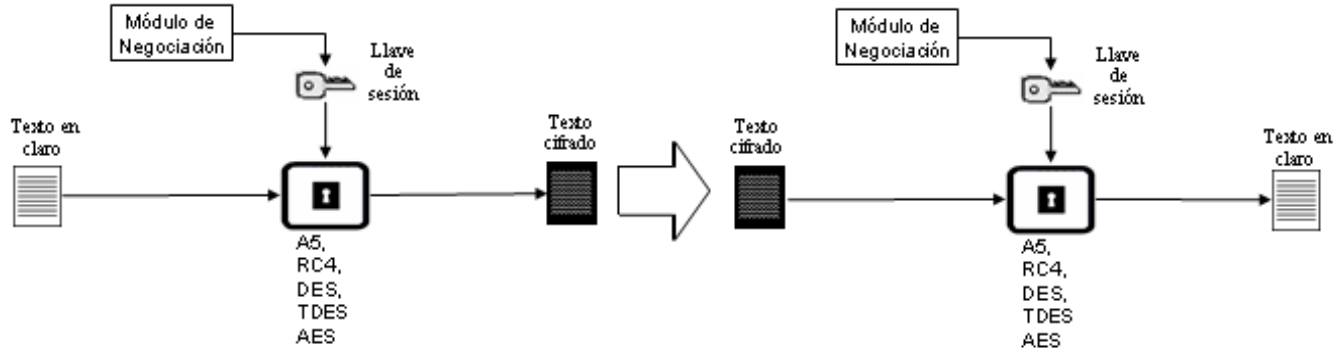

Figura 5.3: Esquema de cifrado/descifrado del sistema

AES.

Para el caso de la firma y verificación digital el sistema se diferencia de PGP en que usa SHA-1 para obtener la huella digital o digesto y además porque se puede usar también ECC como criptosistema de llave pública. La figura 5.4 muestra el esquema de firma/verificación digital del sistema desarrollado.

En la siguiente sección se muestra con mayor detalle la arquitectura y los componentes del proyecto desarrollado.

5.1.2. Arquitectura del sistema

La arquitectura del sistema se muestra en la figura 5.5. Para analizar ésta se describirán los subsistemas cliente y servidor, así como la comunicación entre ambos.

Como ya se mencionó, de forma general el sistema está construido siguiendo el paradigma cliente servidor. De manera particular, ambas entidades basan su funcionamiento en el modelo entrada-proceso-salida. Dicho modelo muestra la interacción entre las diversas partes del sistema con base a las entradas que se procesan y las salidas obtenidas de dicho proceso [20].

Con base a este modelo a continuación se presenta para cada una de las entidades las entradas, los procesos y las salidas que interviene en el sistema.

Cliente.

Las entradas se pueden definir como la información que el usuario o un proceso provee al sistema para

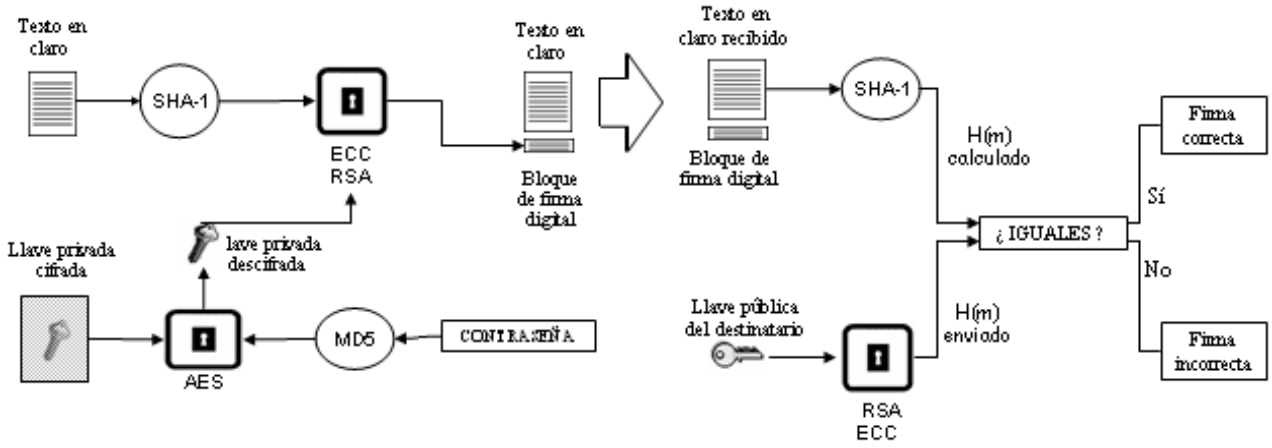

Figura 5.4: Esquema de firma/verificación digital del sistema

que sea procesada. Para el caso del cliente, la información que se alimenta al sistema proviene de dos fuentes: por medio de la interfaz gráfica y por un puerto de red.

1. Entrada por interfaz gráfica: Por medio de la interfaz gráfica desarrollada, se obtienen los parámetros criptográficos que el protocolo de negociación requiere, los nombres de los archivos que se van a intercambiar y la contraseña del usuario para descifrar su llave privada. La información que entra por este medio es exclusivamente otorgada por el usuario.

2. Entrada por comunicación inalámbrica: La información entra por un puerto de red, dicho puerto es creado al entablar la comunicación entre ambos dispositivos. La información proviene del servidor, y es producto de algún proceso. Entre los datos que son enviados están todos los mensajes involucrados en el proceso de negociación (HolaServidor, certificado digital, petición de certificado, especificación de cambio de cifrador), y los datos intercambiados de forma segura (datos cifrados, mensajes firmados digitalmente, peticiones de intercambio).

El cliente engloba sus procesos en dos módulos principales, los cuales son:

1. Módulo de Negociación: Este módulo consta de procesos que tienen como objetivo establecer una sesión segura, es decir, crear una llave de sesión usando criptografía de llave pública y protocolos de

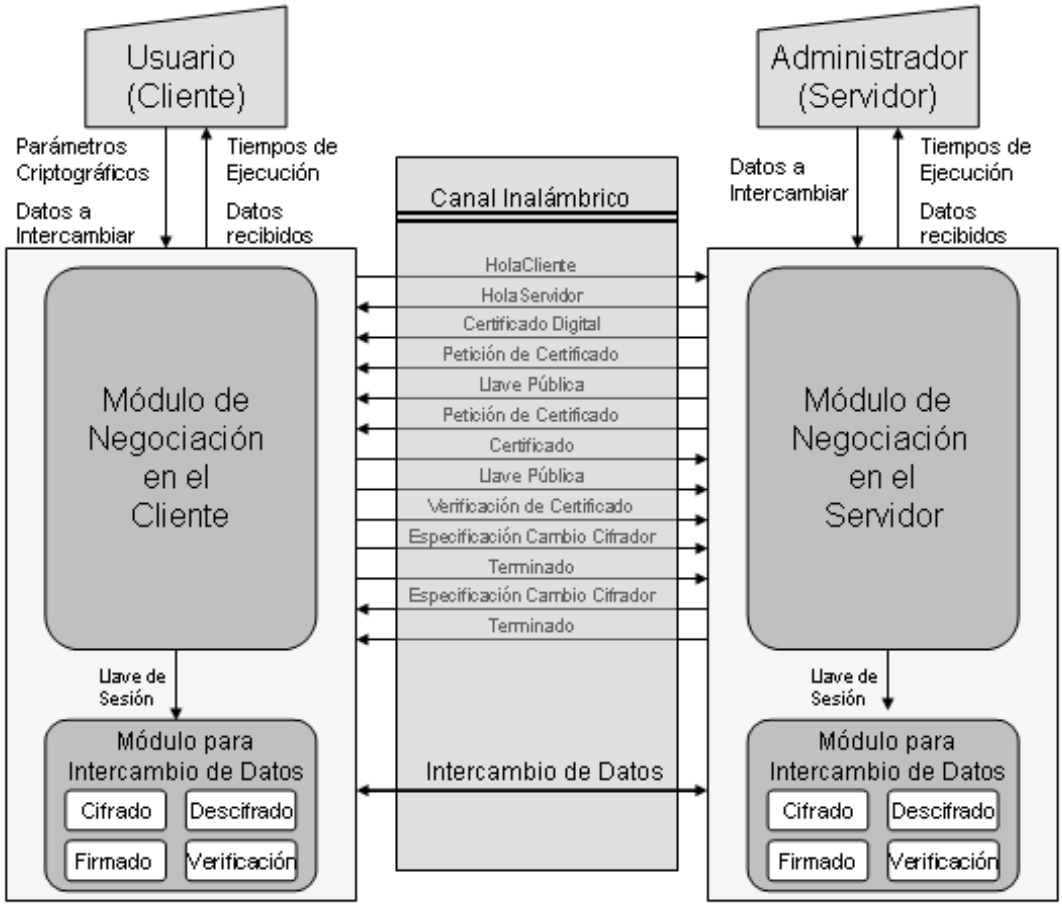

Figura 5.5: Arquitectura del sistema de seguridad para intercambio de datos para dispositivos móviles

seguridad, como los descritos en la sección 4.3. Entre los procesos más importantes en este modulo están: creación de mensaje HolaCliente, preparar llave pública, analizar certificado, preparar certificado y calcular llave de sesión.

2. Módulo para Intercambio de Datos: Dentro de este módulo se tiene cuatro procesos principales, que son: cifrar y descifrar datos y firma/verificación digital. Con dichos procesos se puede llevar a cabo un intercambio seguro de información basado en la confidencialidad, autenticación e integridad.

La información procesada, es decir, la salida del sistema puede ser dirigida a un puerto de red o a un archivo, ambos tipos de salida se describen a continuación:

1. Salida por comunicación inalámbrica: La información se manda al servidor por un puerto de red. Los datos que son enviados son los mensajes involucrados en el proceso de negociación (HolaCliente, certificado digital, especificación de cambio de cifrador, llave pública), y los datos intercambiados (datos cifrados, archivos firmados, peticiones de envío de archivos).

2. Salida a archivo. Los datos que son descifrados o que son verificados digitalmente se almacenan en archivos.

Cabe señalar que la interfaz gráfica por medio de mensajes avisa al usuario de irregularidades durante el proceso o de la culminación exitosa de transacciones.

A continuación se describen las entradas, los proceso y las salidas del servidor.

Servidor.

Para el caso del servidor casi toda la información que se alimenta a los procesos proviene del cliente, a excepción de la contraseña del administrador para el caso de firmado digital de mensajes. A continuación se describen brevemente las fuentes de información para el servidor.

1. Entrada por consola: Por medio de la consola se acepta la contraseña del administrador del servidor para descifrar su llave privada si desea firmar digitalmente un archivo.

2. Entrada por comunicación inalámbrica: La información que envía el cliente entra por un puerto de red. Entre los datos que se reciben por este medio están todos los mensajes involucrados en el proceso de negociación (HolaCliente, certificado digital, llave pública) y los datos intercambiados con el cliente (datos cifrados, archivos firmados digitalmente, peticiones de intercambio de datos).

El servidor también engloba los procesos en módulos principales, los cuales son:

1. Módulo de Negociación: Con este módulo se busca establecer una sesión segura usando criptografía de llave pública y protocolos de seguridad. Los procesos más importantes en este módulo son: extraer y analizar HolaCliente, crear mensaje HolaServidor, preparar llave pública, preparar certificado, analizar certificado, calcular llave de sesión.

2. Módulo para Intercambio de Datos: El objetivo de este módulo es intercambiar información con el cliente de forma segura. Consta de cuatro procesos principales los cuales son: los procesos para cifrar y descifrar datos y los procesos para firma/verificación digital.

La salida del sistema en el servidor es dirigida al cliente por medio de un puerto de red y una red inalámbrica o a archivos.

1. Salida a archivos: El servidor registra los tiempos de ejecución y los almacena en archivos para que el administrador los analice y así obtener información acerca del rendimiento del sistema. También los datos descifrados o verificados digitalmente se almacenan en archivos.

2. Salida por comunicación inalámbrica: La información que ha sido procesada es dirigida al cliente por un puerto de red. Los datos que se envían por este medio son los mensajes usados en el protocolo de negociación (HolaServidor, certificado digital, llave pública, especificación de cambio de cifrado), y los datos que se comparten con el cliente (datos cifrados, mensajes firmados, aceptación de peticiones).

Para el caso del servidor no se cuenta con una interfaz gráfica, por lo que cualquier aviso sobre irregularidades durante la sesión o avisos de transacciones exitosas se despliegan en una consola.

En el siguiente apartado se describe, con ayuda de diagramas de flujo los módulos y procesos involucrados de ambos sistemas.

5.1.3. Diagramas y especificación de procesos.

En esta sección se describen de forma detallada los procesos más importantes del sistema. Primero se presenta la descripción del MN del cliente y del servidor. La funcionalidad del MN se refiere en términos más específicos a lo explicado en la sección 4.3.4. Posteriormente se describe el MID de ambas entidades.

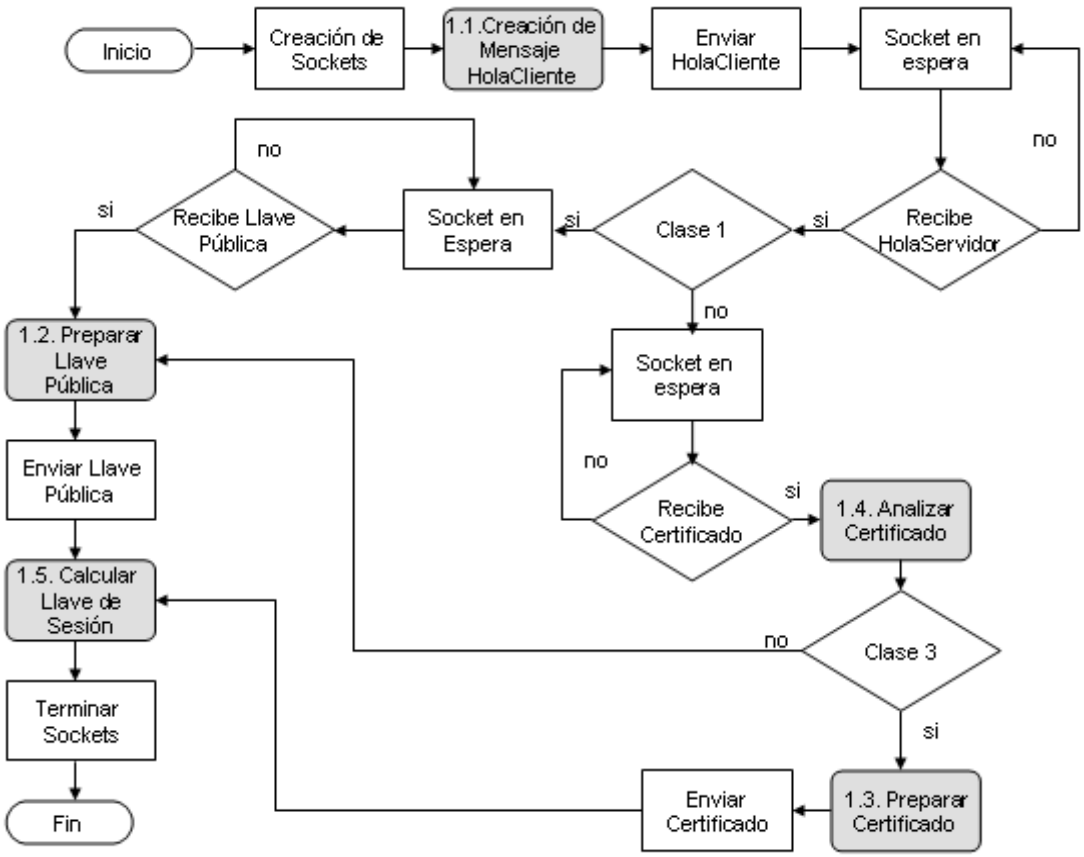

Figura 5.6: Diagrama de flujo del módulo de negociación para el cliente

Módulo de Negociación en el cliente.

El diagrama de flujo para el Módulo de Negociación del cliente se muestra en la figura 5.6.

Después de obtener los paramétros criptográficos vía interfaz gráfica y de entablar la comunicación inalámbrica por medio de sockets se ejecuta el MN. Lo primero que se hace en este módulo es crear un mensaje denominado HolaCliente con las especificaciones hechas por el usuario. Los parámetros que se utilizan para crear el mensaje HolaCliente son: la definición de los algoritmos de firma digital y de acuerdo de llave y las opciones de los parámetros a utilizarse con base al estándar WTLS [19]. Dicho mensaje se envía en una estructura de almacenamiento temporal (buffer) por el canal de comunicación hacia el

servidor. Ya que se ha enviado el mensaje inicial, el cliente va a esperar un mensaje del servidor, denominado HolaServidor, el cual confirma la petición y la utilización de los parámetros elegidos por el usuario. Como WTLS tiene tres tipos de implementación, los pasos a seguir varían con respecto al tipo de clase que se eligió:

- Clase 1. Dado que no existe una autenticación entre las entidades, el cliente genera en este momento, la llave pública y la llave privada para la sesión actual. Todas estas variables se generan a partir de los parámetros criptográficos que el usuario eligio y que fueron confirmados por el servidor. El cliente, después de generar el par de llaves, espera a que el servidor le envíe su llave pública, al recibirla, envía su propia llave pública.

- Clase 2. En esta implementación existe autenticación por parte del servidor, por ello, el cliente espera que éste envíe su certificado, al recibirlo, se verifica la autenticidad del mismo y se extrae la llave pública del servidor. Posteriormente se preparan las llaves pública y privada para la sesión actual, generadas a partir de los parámetros criptográficos que el cliente eligió y que fueron confirmados por el servidor. Finalmente el cliente envía su llave pública al servidor.

- Clase 3. En esta implementación hay autenticación de las dos partes, por lo que el cliente espera el certificado del servidor, una vez recibido verifica su autenticidad y extrae la llave pública de éste. A su vez, el cliente prepara y envía su certificado para que el servidor lo autentique y pueda obtener su llave pública. El cliente recupera su llave privada de un repositorio.

Al mismo tiempo en el servidor se ejecuta el Módulo de Negociación el cual se describe a continuación.

Módulo de Negociación en el servidor

El diagrama de flujo para el Módulo de Negociación para el servidor se muestra en la figura 5.7. A diferencia del cliente, el servidor no recibe datos iniciales por parte del usuario, sino del dispositivo cliente. El cliente comienza la comunicación al enviar el mensaje HolaCliente. En dicho mensaje va la propuesta del cliente para iniciar una sesión segura y los parámetros elegidos para la creación de la misma. El servidor verifica que dichos parámetros sean válidos, si lo son, se crea el mensaje HolaServidor con el cual se confirma la petición de crear una sesión segura con base en los parámetros especificados. Dependiendo de la clase de implementación de WTLS elegida se pueden tener los siguientes casos:

- Clase 1. El servidor genera las llaves pública y privada para ser usadas en la sesión. Las variables

requeridas para llevar a cabo el protocolo de negociación se crean a partir de los parámetros criptográficos elegidos por el cliente. El servidor envía su llave pública y espera la llave pública del cliente.

- Clase 2. En este caso, el servidor recupera el certificado que corresponde a los parámetros especificados por el cliente y lo envía. Del certificado se extrae la llave pública. Asimismo, se recupera la llave privada correspondiente a la llave pública extraída del certificado. Finalmente, el servidor espera a que el cliente le envíe su llave pública.

- Clase 3. El servidor obtiene el certificado que corresponde a los parámetros indicados por el cliente y lo envía. Del certificado se extrae la llave pública. Se recupera la llave privada que corresponde a la obtenida del certificado. El servidor manda al cliente una petición de envío de certificado. Cuando el servidor recibe el certificado del cliente, verifica la autenticidad del mismo y extrae la llave pública que corresponde al cliente.

Cuando ambas entidades poseen las llaves públicas de su contraparte generan la llave de sesión que será utilizada en el MID.

Especificación de procesos para Módulo de Negociación.

Los módulos descritos anteriormente están conformados por distintos procesos, tal como se muestra en los diagramas de flujo mostrados en las figuras 5.6, 5.7. A continuación se muestra la descripción de cada uno de ellos.

- Proceso 1.1 *Creación de Mensaje HolaCliente* toma como parámetros de entrada las opciones que el usuario eligió e inicializa los campos de la estructura ClientHello [19]. Para el caso del campo *key exchange suite*, el cual verifica el tipo de intercambio de llave, se evalúa lo siguiente:

```
switch(key_exchange_suite){
    case ECDH_anon:
    case ECDH_oneway:
        leer los parámetros de la curva del archivo;
        el tamaño del certificado del servidor es 0;
        se asigna como AC a CINVESTAV_CA;
        break;
    case RSA_anon:
```

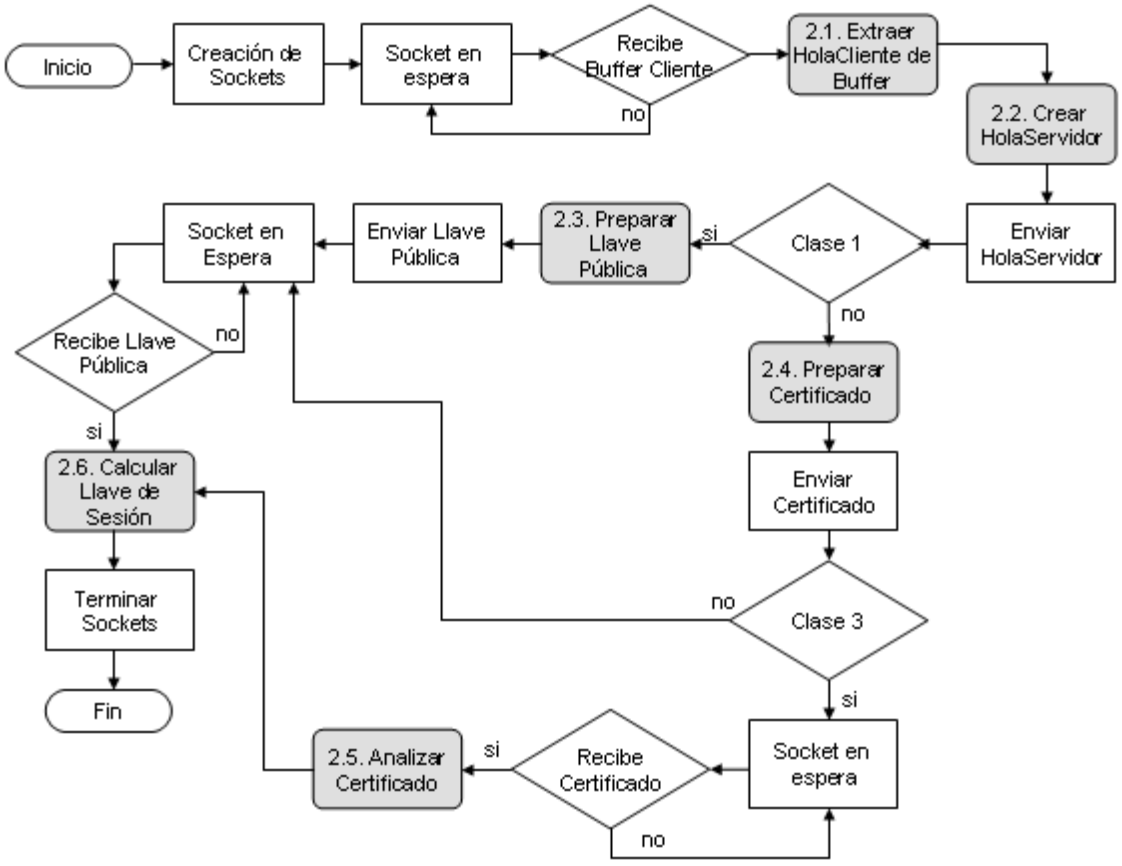

Figura 5.7: Diagrama de flujo de módulo de negociación para el servidor

case RSA_oneway:
se asignan los parámetros necesarios para RSA;
el tamaño del certificado del servidor es 0;
se asigna como AC a CINVESTAV_AC;
break;
case ECDH_twoway:
se abre el certificado correspondiente;
se asigna como AC a CINVESTAV_AC;
break;
case RSA_twoway:
se abre el certificado correspondiente;
se asigna como AC a CINVESTAV_AC;
break;
case ECDH_twowayHYBRID:
se abre el certificado correspondiente;
se asigna como AC a CINVESTAV_AC;
break;
case NULLKES:
default:
no habrá generación de secreto
}

Cabe señalar que para este tema de tesis se añadió una implementación híbrida la cual se discutirá más adelante.

- Proceso 2.1 *Extraer HolaCliente de Buffer*. El servidor recibe dentro de un buffer el mensaje HolaCliente de forma compacta, por lo tanto, debe extraer la información para llenar la estructura ClientHello.

- Proceso 2.2 *Crear HolaServidor*. A partir del mensaje HolaCliente el servidor verifica que los parámetros que se indican en dicho mensaje sean correctos. Una vez verificada la información, se genera el mensaje HolaServidor completando la estructura ServerHello [19].

- Procesos 1.2 y 2.3 *Preparar Llave Pública*. Para crear el par de llaves, es decir, la pública y la privada ambas entidades realizan lo siguiente:

```
switch(key_exchange_suite){
    case ECDH:
        iniciar la memoria necesaria para almacenar las llaves;
        generar la llave pública y privada para curvas elípticas;
        break;
    case RSA:
        iniciar la memoria necesaria para almacenar las llaves;
        generar la llave pública y privada para RSA
        break;
    case NULLKES:
        default:
            no hay generación de secreto
}
```

- Procesos 1.3 y 2.4 *Preparar Certificado*. Con base en el sistema de llave pública especificado en HolaCliente se recupera el certificado, el cual está almacenado en disco, se asigna la información a la estructura *SignatureInfo [19]* y se extrae la llave pública del usuario del certificado.

- Procesos 1.4 y 2.5 *Analizar Certificado.* Con base en la estructura y descripción de un certificado vistos en la sección 4.2 se realiza el análisis del mismo. Inicialmente se extraen los siguientes campos:

1. La firma digital del certificado.
2. El identificador del algoritmo de firma digital.
3. El identificador de la autoridad certificadora que lo firmó.

Con dicha información se realiza la verificación del certificado con el algoritmo indicado (PKCS-RSA ó ECDSA) y con la llave pública de la autoridad certificadora.

Si el certificado es verificado, se extrae lo siguiente:

1. La llave pública del usuario.
2. El identificador del algoritmo de generación de llave de sesión (RSA o ECDH).

- Procesos 1.5 y 2.6 *Calcular Llave de Sesión*. Para generar la llave de sesión, dependiendo del algoritmo elegido se realiza lo siguiente:

```
switch(key_exchange_suite){
    case ECDH:
        generar llave de sesión usando curvas elípticas;
        break;
    case RSA:
        generar llave de sesión para curvas RSA;
        break;
    case NULLKES:
    default:
        no habrá generación de llave
}
```

Ya que se ha descrito el MN tanto en el cliente como en el servidor, lo siguiente es detallar el MID en ambas entidades.

Módulo para Intercambio de Datos en el Cliente.

El objetivo principal de este módulo es mantener el intercambio de información mientras que la sesión esté activa. El diagrama de flujo que describe a este módulo se muestra en la figura 5.8. Como se puede observar en dicha figura, en el MID se evalúa el tipo de transacciones que el usuario desea llevar a cabo. Dentro de las opciones que se presentan al usuario vía interfaz gráfica son:

- Cifrar Datos. Cuando el cliente desea enviar un documento de manera confidencial debe informárselo al servidor. Si su petición es aceptada, se elige el archivo a enviar y se crea una estructura denominada cabecera, la cual contiene información necesaria para crear el archivo en el servidor (nombre del archivo, extensión, tamaño). Una vez enviada la cabecera, el contenido del archivo se copia a un buffer para posteriormente cifrarlo y enviarlo por el canal inalámbrico.

- Descifrar Datos. Para recibir un archivo por parte del servidor, el cliente hace una petición para que le sea enviada una lista de los archivos que posee éste. Elige uno de los archivos y hace la petición para que se envíe ese archivo de forma confidencial. El servidor manda la cabecera del archivo y un

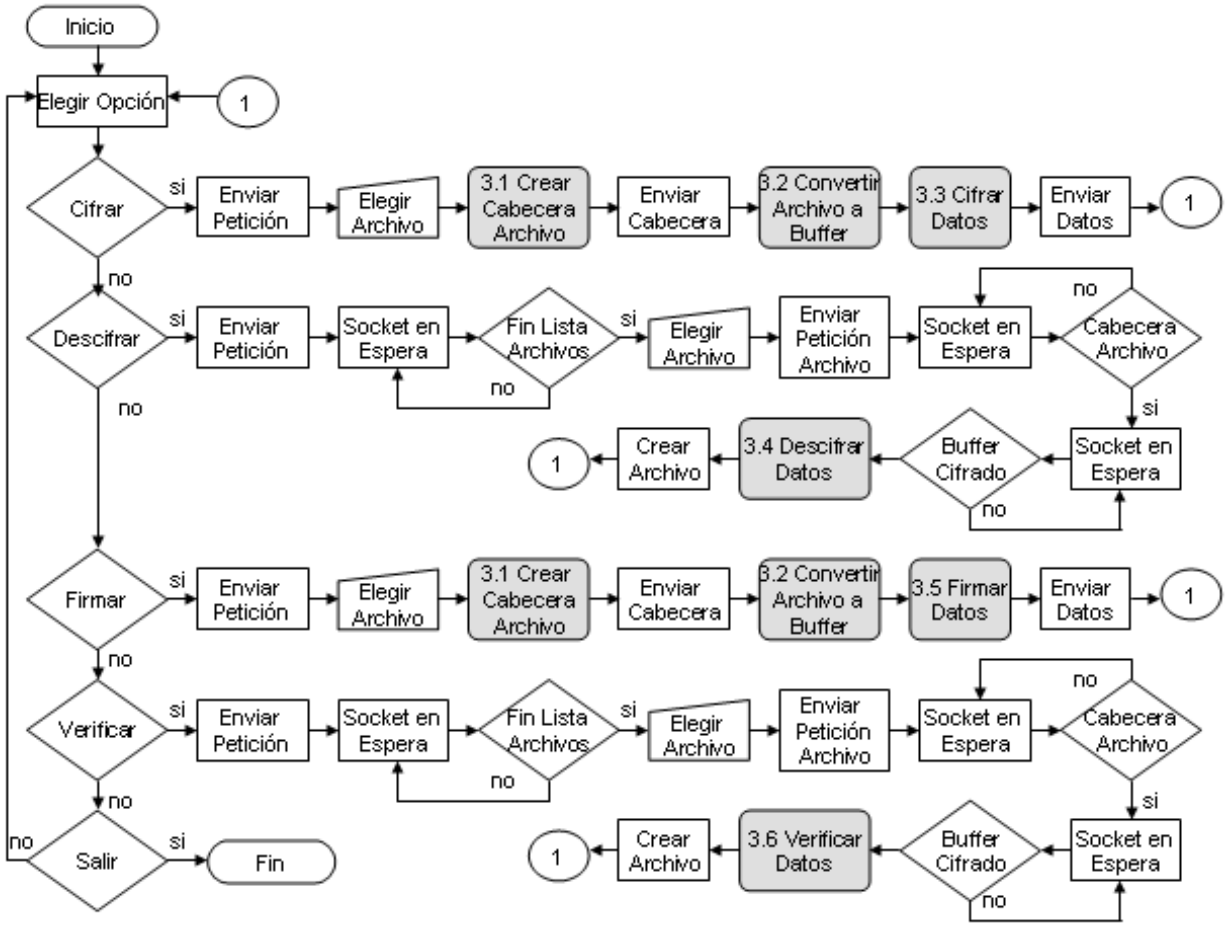

Figura 5.8: Diagrama de flujo del módulo para intercambio de datos en el cliente

buffer con los datos cifrados. Cuando se reciben esos datos, el buffer se hace pasar por el proceso de descifrado y se crea el archivo.

- Firmar Datos Digitalmente. El cliente debe de hacer una petición de envío de datos firmados al servidor. Si su petición es aceptada, el cliente elige el archivo que va a firmar digitalmente, crea su cabecera y la manda al servidor. El contenido del archivo se copia a un buffer para obtener su digesto usando una función hash. Dicho digesto se firma digitalmente. Tanto el buffer que contiene la información del archivo como la firma digital son enviados al servidor.

- Verificar Datos Digitalmente. Para recibir un archivo firmado digitalmente por el servidor, el cliente le hace una petición para que le sea enviada una lista de los archivos que puede compartir éste. Elige uno y realiza una petición para que dicho archivo le sea regresado firmado digitalmente. El servidor manda la cabecera del archivo, el contenido del mismo en un buffer y la firma digital. Tanto el buffer como su firma digital son tomados en cuenta en el proceso de verificación digital para corroborar que la firma del archivo sea válida. Si la firma es válida se crea el archivo, en otro caso, el archivo no es creado.

Para atender las peticiones de intercambio de datos, el servidor cuenta con su propio MID el cual se detalla a continuación.

Modulo para Intercambio de Datos en el servidor.

El diagrama de flujo que describe a este módulo se muestra en la figura 5.9. Para que el servidor atienda las peticiones de intercambio de información del cliente es necesario que éste le envíe el tipo de transacción que desea llevar a cabo.

- Cifrar Datos. Cuando el cliente desea que el servidor le envíe un documento cifrado le pide a éste una lista de todos los archivos disponibles para intercambiar. El servidor manda dicha lista y espera la petición de envío de un archivo específico. Cuando se recibe el nombre del archivo, se crea su cabecera y se remite al cliente. Posteriormente, se extraen los datos del archivo a un buffer, este último se cifra utilizando el algoritmo previamente acordado, la salida se envía por el canal inalámbrico.

- Descifrar Datos. El servidor recibe por parte del cliente una petición para recibir un archivo cifrado, acepta la petición y se prepara para recibir la cabecera del archivo y los datos. Cuando son recibidos se hace pasar por el proceso de descifrado. Finalmente se crea el archivo con los datos de la cabecera (nombre, extensión, tamaño) y con la salida del proceso de descifrado.

- Firmar Datos Digitalmente. Cuando el cliente desea que el servidor le envie un documento firmado digitalmente, hace una petición para que se le envíe una lista de todos los archivos que el servidor tenga disponibles para intercambiar. El servidor envía dicha lista y espera que el cliente regrese el nombre del archivo elegido. Cuando se recibe el nombre del archivo, se crea la cabecera del mismo y se envía al cliente. Posteriormente se extraen los datos del archivo a un buffer, este último se firma digitalmente. Finalmente los datos y la firma digital se envian por el canal inalámbrico.

- Verificar Datos Digitalmente. El servidor recibe por parte del cliente una petición para recibir un archivo firmado digitalmente. El servidor acepta la petición y se prepara para recibir la cabecera del archivo. Después de recibir la cabecera, el servidor espera recibir los datos y la firma digital. Una vez que se reciben se hacen pasar por el proceso de verificación para comprobar la autenticidad de la firma. Si la verificación es exitosa, se crea el archivo con los datos de la cabecera y con el buffer que contiene los datos del archivo, si no, no es creado.

Especificación de procesos para el Módulo para Intercambio de Datos.
A continuación se muestra la descripción de los procesos que conforman a cada MID.

- Procesos 3.1 y 4.1 *Crear Cabecera Archivo*: El objetivo de este proceso es crear una estructura denominada Header. Con dicha cabecera se puede crear un archivo que fue enviado cifrado o firmado en el dispositivo receptor. Dicha estructura posee el nombre del archivo, su extensión, su tamaño y el número de paquetes en los que se puede dividir.

- Procesos 3.2 y 4.2 *Convertir Archivo a Buffer*: En este proceso se extrae el contenido de un archivo y se copia en un buffer. Dicha estructura es más fácil de manipular por los procesos que un archivo.

- Procesos 3.3 y 4.3 *Cifrar Datos*: Al iniciar el proceso para cifrar datos se verifica qué algoritmo de cifrado se va a utilizar. Dicho algoritmo es uno de los parámetros que el cliente eligió al inicio de la sesión. En este proceso se hace uso de la llave de sesión creada en el MN. Los algoritmos de cifrado disponibles son el: A5, RC4, DES, TDES, AES (usando llaves de 128 y 256 bits). Dichos algoritmos están implementados en la biblioteca criptográfica desarrollada en [21]. La figura 5.10 (a) muestra el diagrama del proceso para cifrar de datos.

- Procesos 3.4 y 4.4 *Descifrar Datos*: Lo primero que se realiza dentro de este proceso es verificar qué algoritmo de cifrado se acordó. Identificado el algoritmo, se hace uso de la llave de sesión y se obtiene

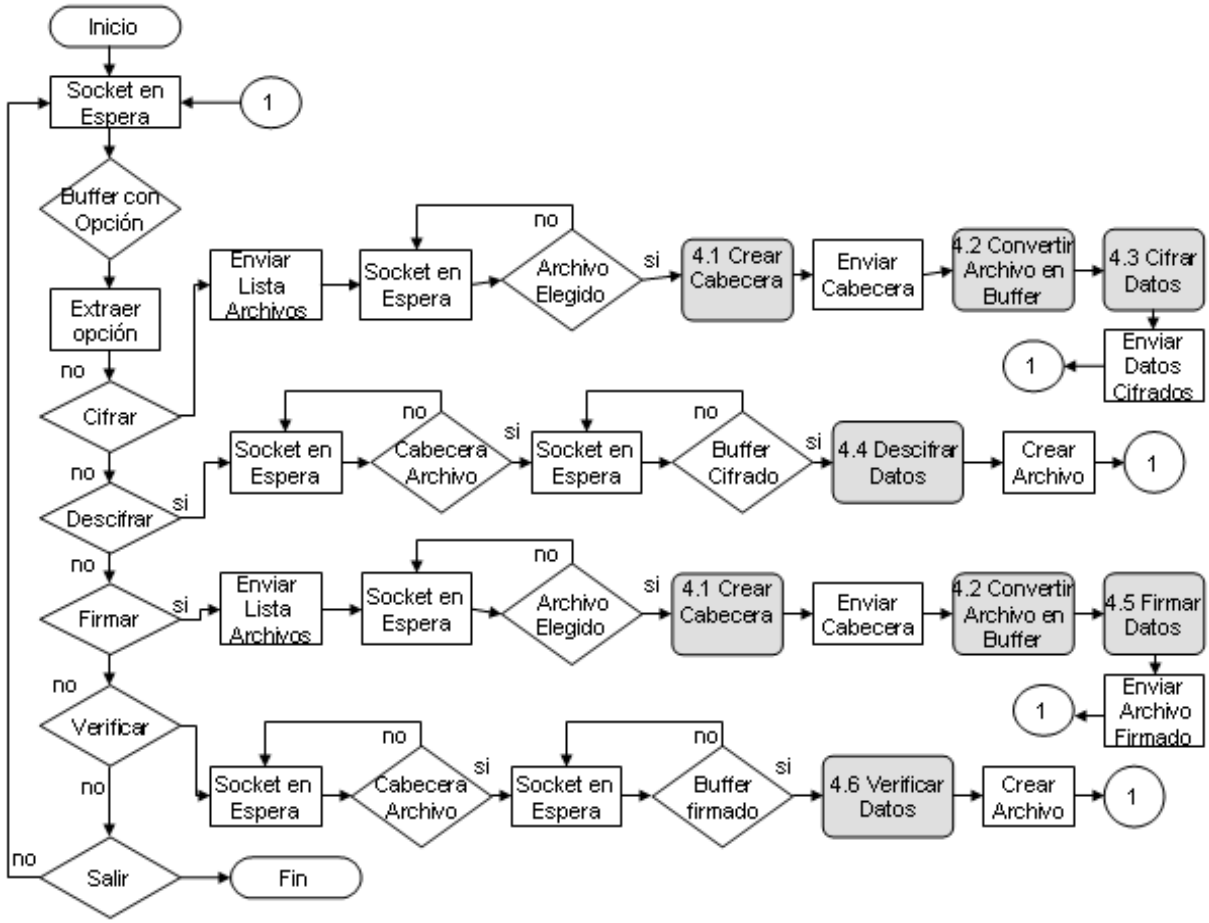

Figura 5.9: Diagrama de flujo de módulo para intercambio de datos en el servidor

el texto en claro. La figura 5.10 (b) muestra el diagrama del proceso para descifrar datos.

- Procesos 3.5 y 4.5 *Firmar Datos*: Para firmar datos digitalmente, se evalúa qué algoritmo se eligió y por consiguiente, qué algoritmo de llave pública se está usando. Dado que el protocolo de negociación de WTLS sólo acepta dos criptosistemas de llave pública, en la etapa de intercambio de datos se mantiene esa premisa. Por lo tanto, sólo se podrá firmar utilizando ECDSA o RSA usando el algoritmo SHA-1 como función hash. Para llevar a cabo la firma digital se utiliza la llave privada de la entidad que va a firmar. La figura 5.11 (a) muestra el diagrama correspondiente al proceso de Firmar Datos.
- Procesos 3.6 y 4.6 *Verificación de Datos*: Cuando se desea verificar una firma digital inicialmente se identifica qué algoritmo para firmar se estableció en la negociación y por consiguiente qué algoritmo de llave pública se usó. Una vez que se identificó el algoritmo, se hace pasar el mensaje y la firma digital por la función correspondiente de verificación digital. Para verificar una firma es necesario utilizar la llave pública de la entidad que emitió la firma digital. La implementación de las funciones de firma y verificación digital con ECC y RSA fueron implementadas en la biblioteca criptográfica desarrollada en [21]. El diagrama que describe el proceso de Verificación digital se muestra en la figura 5.11 (b).

En la siguiente sección se muestran algunos detalles de implementación.

5.2. Detalles de Implementación del sistema.

El Sistema de Seguridad para Intercambio de Datos en Dispositivos Móviles está implementado en ANSI C, para el Módulo de Negociación se siguieron las especificaciones del estándar WTLS [19].

Para las operaciones criptográficas se utilizó el trabajo descrito en [21], el cual fue realizado en la Universidad de Oregon State. En dicho trabajo se desarrolló un conjunto de herramientas criptográficas, englobadas en una biblioteca conocida como RCT. Entre los algoritmos incluidos en la biblioteca se encuentran implementaciones de criptografía de llave pública con RSA y Curvas Elípticas; RC4, AES y DES para la criptografía de llave simétrica, y la familia SHA para las funciones hash, adicionalmente, dicha biblioteca contiene un prototipo del protocolo de negociación de WTLS.

De manera general el procedimiento seguido en el desarrollo del Sistema de Seguridad para Intercambio de Datos en Dispositivos Móviles fue el siguiente:

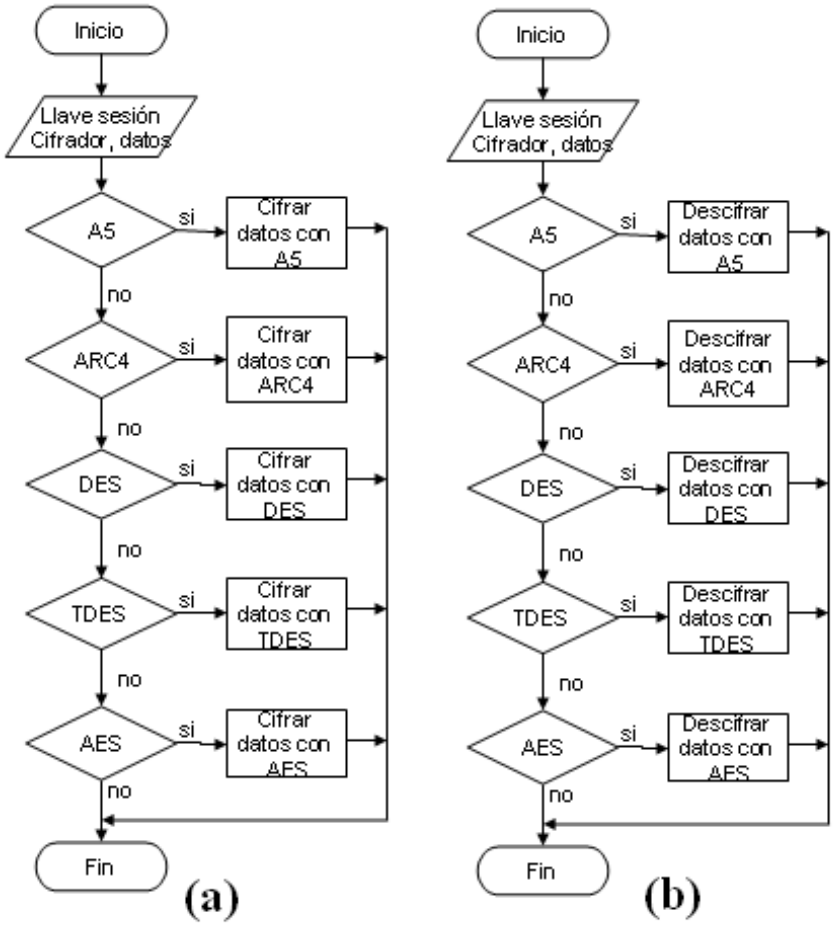

Figura 5.10: (a) Proceso para cifrar datos, (b) Proceso para descifrar datos

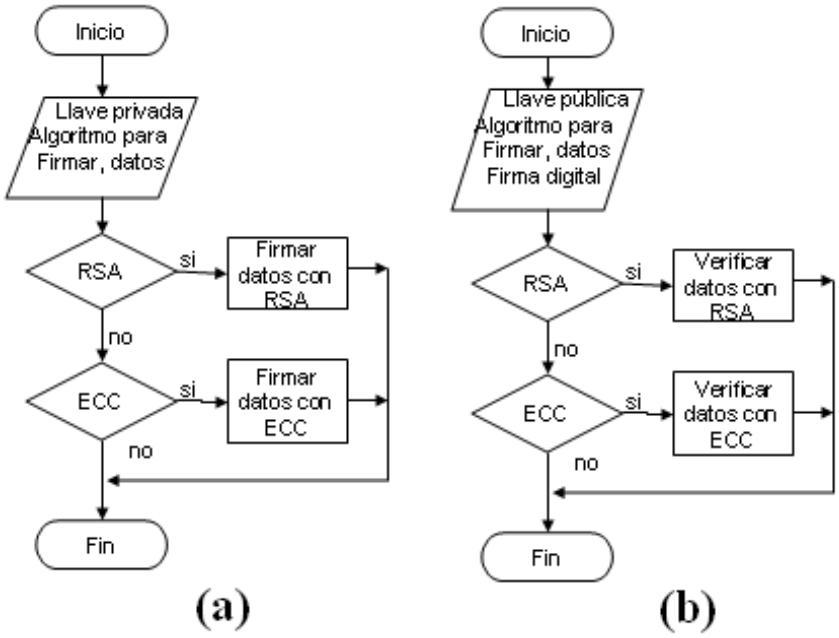

Figura 5.11: (a) Proceso para firmar digitalmente, (b) Proceso para verificación digital

1. Migración de la biblioteca criptográfica y del protocolo de negociación a un dispositivo móvil ligero.
2. Implementación de un esquema híbrido al protocolo de negociación de WTLS.
3. Desarrollo de el MID para ambas plataformas, es decir, cliente y servidor.

En las siguientes subsecciones se describe brevemente cada uno de los puntos anteriores:

5.2.1. Migración de la biblioteca criptográfica y del protocolo de negociación a un dispositivo móvil ligero.

En este trabajo de tesis se utilizó un dispositivo móvil ligero, específicamente un PDA Compac IPAQ PC Modelo H3950, como cliente. Para realizar dicha migración se consideraron los siguientes aspectos:

- Sistema Operativo. La IPAQ h3950 trae pre-instalado el sistema operativo Windows Powered. Dicho sistema operativo soporta la comunicación inalámbrica gracias a una librería de comunicación por sockets (Winsocks).
- Compilador. El compilador Embedded Visual C++ 3.0 está disponible para el desarrollo de aplicaciones no comerciales. Dadas las similitudes entre este compilador y el Visual C++ 6.0 (compilador con el cual fue desarrollada la biblioteca criptográfica), fue elegido para el desarrollo del sistema en el dispositivo móvil.

Al término de la migración de la biblioteca criptográfica y del protocolo de negociación se hicieron pruebas para verificar el buen funcionamiento de los algoritmos criptográficos y, posteriormente, se hicieron pruebas de rendimiento al protocolo de negociación, los resultados obtenidos de dichas pruebas se discutirán en el siguiente capítulo.

5.2.2. Implemententación de esquema híbrido.

La modificación más importante al protocolo de negociación que se realizó en este trabajo fue implementar una esquema híbrido donde se conjuntaron las ventajas de los dos criptosistemas que soporta el protocolo de negociación de WTLS, es decir, ECC y RSA.

Dicho cambio se refleja en la definición de dos estructuras de datos del estándar. La primer estructura modificada especifica el tipo de intercambio de llave y el tipo de autenticación a utilizarse. La descripción más detallada de las estructuras se puede consultar en [19].

KeyExchangeSuite{

NULLKES, ECDH_anon, RSA_anon, ECDH_oneway, RSA_oneway,
ECDH_twoway, RSA_twoway, **ECDH_twowayHYBRID**

}

1. La segunda estructura que fue modificada especifica el algoritmo para realizar la firma digital. La estructura modificada quedó como sigue:

SignatureAlgorithm{

ANONYMOUS, ECDSA_SHA1, RSA_SHA1, **HYBRID_RSA_SHA1**

}

El objetivo principal del modelo híbrido es mejorar el rendimiento del protocolo de negociación de WTLS. Dicho modelo está enfocado exclusivamente a la clase de implementación 3 del protocolo, es decir, cuando hay autenticación mutua. El modelo híbrido realiza la verificación de los certificados digitales, tanto del cliente y del servidor, usando RSA, para ello se crearon certificados digitales apegados al estándar X.509 donde se combinan llaves de ECC y de RSA. En otras palabras, el certificado se firma usando RSA, pero la llave de la entidad certificada se generó con ECC. La descripción con base en las operaciones criptográficas que se realizan en el esquema híbrido se muestran en la sección 6.1.

5.2.3. Desarrollo del MID para ambas plataformas.

Se implementó el MID con base en el diseño previamente expuesto para cada una de las plataformas usadas en este proyecto. Algunos de los procesos descritos en la sección 5.1.3 hacen uso de los algoritmos criptográficos A5, RC4, DES, TDES, AES para el caso de cifrar/descifrar datos, para funciones hash hacen uso de los algoritmos MD5 y SHA1 y finalmente para firmar/verificar digitalmente se usan RSA y ECC. Todos ellos pertenecientes a la biblioteca criptográfica, excepto el algoritmo A5, el cual se agregó como parte de este proyecto.

Información detallada acerca del funcionamiento del sistema se presenta en el Apéndice A.

Capítulo 6

Análisis de Desempeño

Las restricciones computacionales de los dispositivos móviles y las carencias que aún tienen las redes inalámbricas (un ancho de banda restringido y una latencia relativamente alta) obligan a que para lograr una implementación eficiente del protocolo de negociación de WTLS es necesario que el tiempo de procesamiento no sea muy alto y que los mensajes que se intercambian no sean muy grandes.

En [22], se presenta un estudio analítico del desempeño de los dos criptosistemas de llave pública utilizados por WTLS. Allí fue concluido de manera teórica que ECC tiene un mejor desempeño que RSA. En [23] se realizó un análisis comparativo de los sistemas criptográficos ejecutando al cliente y al servidor dentro de una misma computadora. En [24] se llevó a cabo un análisis de desempeño en SSL donde ECC superó a RSA en vario aspectos.

La contribución de este trabajo de tesis es presentar una primera aproximación hacia un escenario inalámbrico realista, donde las predicciones del modelo analítico se puedan confirmar con los datos reales obtenidos de la puesta en práctica del protocolo de negociación de WTLS. La evidencia experimental demuestra que ECC rinde un funcionamiento mejor que la opción tradicional representada por RSA en todos los escenarios analizados.

El resto de éste capítulo está organizado como sigue, en la sección 6.1 se describen las pruebas realizadas, en la parte 6.2 se muestran los resultados obtenidos y finalmente se hace un análisis de resultados.

6.1. Pruebas realizadas

El prototipo del protocolo de negociación desarrollado en este trabajo fue implementado en ANSI C siguiendo las indicaciones del estándar WTLS. Para la implementación eficiente de las diversas operaciones criptográficas se utilizó una biblioteca criptográfica desarrollada en [21].

Las plataformas de prueba para los clientes fueron una Compac Ipaq H3950 Intel Xcale a 400 MHz y una laptop HP Pavilion Pentium IV a 2400 GHz, para el servidor se utilizo otra computadora HP Pavilion Pentium IV a 2400 GHz. Para la comunicación inalámbrica se utilizaron sockets de Windows. El tipo de red que se utilizó para las pruebas es una red IEEE 802.11a en su modalidad ad-hoc. Los resultados que se presentan en este capítulo corresponden al experimento de tener dos clientes conectados simultáneamente, los cuales realizarón cientos de negociaciones con el servidor, esto con el fin de obtener un tiempo promedio de duración del protocolo de negociación en cada una de las plataformas.

La tabla 6.1 muestra los tamaños comparables de llaves de ECC y RSA. Esto significa que la seguridad ofrecida por RSA de 1024-bit es comparable con la seguridad ofrecida por ECC con longitudes de llave de 160 y 163 bits que utilizan las curvas predefinidas WTLS 160P, 163K y 163R, respectivamente [19]. De manera semejante, la seguridad asociada a una puesta en práctica de RSA de 2048-bit, se puede obtener usando ECC con longitudes de llave de 224 y 233 bits que utilizan las curvas predefinidas WTLS 224P, 233K y 233R. Los experimentos conducidos en este trabajo consideraban ambos niveles de la seguridad.

Nivel de Seguridad	ECC	RSA
1	160P, 163K, 163R	1024
2	224P, 233K, 233R	2048

Tabla 6.1. Niveles comparables de seguridad para ECC y RSA

Las pruebas realizadas constaron de tomar el tiempo de duración del protocolo de negociación de WTLS. Para obtener dicho tiempo se hicieron cientos de corridas para obtener la media aritmética y así tener un tiempo promedio de duración. Dado que en este proyecto se diseñó un modelo analítico para hacer un estimado del tiempo de duración del protocolo de negociación fue necesario obtener el tiempo de las operaciones criptográficas involucradas en el protocolo, mismas que se listan a continuación:

- Tiempo de ejecución para operaciones públicas usando RSA (cifrado y verificación).
- Tiempo de ejecución para operaciones privadas usando RSA (descifrado y firma).

- Tiempo de ejecución de firmado digital usando ECC.
- Tiempo de ejecución de verificación digital usando ECC.
- Tiempo de ejecución de protocolo Diffie-Hellman usando ECC.

Así mismo se obtuvo el tiempo de:

- Tiempo del estado latente de la comunicación.

En la siguiente sección se reportan los resultados obtenidos de los experimentos realizados a lo largo de este trabajo de tesis.

6.2. Resultados obtenidos

En esta sección se presentan los resultados obtenidos al ejecutar el protocolo de negociación de WTLS para la clase de implementación 3, es decir, con autenticación mutua. Asimismo se presentan los tiempos registrados para cada una de las operaciones criptográficas que fueron utilizados para calcular el tiempo aproximado de dicho protocolo con base al modelo analítico presentado en la sección 4.4. En este estudio se han considerado los dos niveles de seguridad que WTLS soporta, niveles que se han descrito en la tabla 6.1.

Primero se presentan los resultados concernientes al nivel de seguridad 1. La tabla 6.2 muestra el tiempo de ejecución en milisegundos de las operaciones públicas (cifrado-verificación) y privadas (descifrado-firma) de RSA en el nivel de seguridad 1 para las dos plataformas de prueba. La nomenclatura utilizada en las tablas es la descrita en la sección 4.4.

Plataforma	T_{RSA_PUB} Cifrado-Verificación	T_{RSA_PRIV} Descifrado-Firma
HP Pavilion	0.1998	7.7526
IPAQ H3950	1.2194	72.1222

Tabla 6.2. Tiempos de ejecución obtenidos para RSA (en milisegundos) para nivel de seguridad 1.

Las tablas 6.3, 6.4, 6.5 muestran el tiempo de ejecución de las operaciones ECDH, ECDSA firma digital y ECDSA verificación digital para ECC respectivamente.

Plataforma	Curva 160 P T_{ECDH}	Curva 163 K T_{ECDH}	Curva 163 R T_{ECDH}
HP Pavilion	0.115477	0.472773	0.586424
IPAQ H3950	0.70189	3.496	3.529

Tabla 6.3. Tiempos de ejecución obtenidos para ECDH nivel 1.

Plataforma	Curva 160 P T_{ECDSA}(firma)	Curva 163 K T_{ECDSA}(firma)	Curva 163 R T_{ECDSA}(firma)
HP Pavilion	0.19237	0.522003	0.71652
IPAQ H3950	0.78969	4.123	4.348

Tabla 6.4. Tiempos de ejecución obtenidos para ECDSA firma nivel 1.

Plataforma	Curva 160 P T_{ECDSA}	Curva 163 K T_{ECDSA}	Curva 163 R T_{ECDSA}
HP Pavilion	0.40473	1.16212	1.3988
IPAQ H3950	1.6538	8.195	8.70

Tabla 6.5. Tiempos de ejecución obtenidos para ECDSA verificación nivel 1.

La tabla 6.6 muestra el tiempo de ejecución en milisegundos de las operaciones públicas (cifrado-verificación) y privadas (descifrado-firma) de RSA en el nivel de seguridad 2 para las dos plataformas de prueba.

Plataforma	T_{RSA_PUB} Cifrado-Verificación	T_{RSA_PRIV} Descifrado-Firma
HP Pavilion	0.6140	59.8557
IPAQ H3950	7.4007	401.3728

Tabla 6.6. Tiempos de ejecución obtenidos para RSA (en milisegundos) para nivel de seguridad 1.

Las tablas 6.7, 6.8, 6.9 muestran el tiempo de ejecución de las operaciones ECDH, ECDSA firma digital y ECDSA verificación digital para ECC respectivamente.

Plataforma	Curva 224 P T_{ECDH}	Curva 233 K T_{ECDH}	Curva 233 R T_{ECDH}
HP Pavilion	0.249373	0.938250	0.970947
IPAQ H3950	1.503	6.569	6.803

Tabla 6.7. Tiempos de ejecución obtenidos para ECDH nivel 2.

Plataforma	Curva 224 P T_{ECDSA}(firma)	Curva 233 K T_{ECDSA}(firma)	Curva 233 R T_{ECDSA}(firma)
HP Pavilion	0.37321	1.0276	1.2647
IPAQ H3950	2.364	7.401	7.462

Tabla 6.8. Tiempos de ejecución obtenidos para ECDSA firma nivel 2.

Plataforma	Curva 224 P T_{ECDSA}	Curva 233 K T_{ECDSA}	Curva 233 R T_{ECDSA}
HP Pavilion	0.7511	2.1653	2.6739
IPAQ H3950	4.662	14.378	14.724

Tabla 6.9. Tiempos de ejecución obtenidos para ECDSA verificación nivel 2.

En la tabla 6.10 se muestran los tiempos promedio de comunicación registrados en el envío de información entre el cliente y servidor. Cabe señalar que estos tiempos siguen siendo ideales debido a que el escenario inalámbrico con el que se trabajo en esta tesis aún es muy limitado. Si el escenario fuera más dinámico, es decir, si existieran más de dos clientes haciendo peticiones al servidor simultáneamente para establecer una conexión segura, la latencia sería mayor.

$T_{COM}(\mu s)$	k	T_{TCOM}(ms)
9.27	10	0.927

Tabla 6.10. Tiempos de comunicación para el protocolo.

A continuación se muestra el tiempo calculado de duración estimado del protocolo de negociación haciendo uso del modelo propuesto para la clase 3 (autenticación mutua). Es importante mencionar que sólo se calcula el tiempo de duración de las operaciones criptográficas tanto del cliente como del servidor.

En la tabla 6.11 se muestra el tiempo del protocolo de negociación cuando cliente y servidor se encuentran en la misma plataforma.

Hp (Cliente) /HP (Servidor)	T_{HP}(ms)
RSA 1024	22.8532
Curva 160 P	4.1814
Curva 163 K	7.6730
Curva 163 R	8.8217
Híbrido usando curva 160 P	4.5136
Híbrido usando curva 163 K	6.6657
Híbrido usando curva 163 R	7.3579
RSA 2048	139.7652
Curva 224 P	5.7201
Curva 233 K	12.3072
Curva 233 R	14.1477
Híbrido usando curva 224 P	8.8119
Híbrido usando curva 233 K	12.8829
Híbrido usando curva 233 R	13.7194

Tabla 6.11. Tiempo estimado del protocolo de negociación.

Ipaq (Client) / HP (Server)	T_{HP}(ms)
RSA 1024	95.0540
Curva 160 P	6.87932
Curva 163 K	22.6636
Curva 163 R	24.0402
Híbrido usando curva 160 P	6.2398
Híbrido usando curva 163 K	14.7248
Híbrido usando curva 163 R	15.3606
RSA 2048	522.5822
Curva 224 P	13.4501
Curva 233 K	39.0817
Curva 233 R	40.8836
Híbrido usando curva 224 P	16.0517
Híbrido usando curva 233 K	30.5531
Híbrido usando curva 233 R	31.5004

Tabla 6.12 Tiempo estimado del protocolo de negociación.

En la tabla 6.12 se muestra el tiempo del protocolo de negociación cuando cliente y servidor se encuentran en diferente plataforma.

Se observa en las tablas anteriores que el criptosistema de curvas elípticas es una mejor opción para el protocolo de negociación que RSA. Asimismo, el esquema híbrido que se está proponiendo en esta tesis mejora a ECC en los dos niveles de seguridad, con excepción de las curva 160 P y 224 P en ECC.

A continuación se muestra la comparación gráfica entre los tiempos teóricos obtenidos con base en el modelo analítico propuesto y los tiempos experimentales resultado de ejecutar el prototipo del protocolo de negociación. Las figuras 6.1, 6.2 y 6.3 muestran dichos tiempos para el caso de tener como cliente al dispositivo móvil IPAQ H3950.

Las figuras 6.4, 6.5 y 6.6 muestran los tiempos teóricos y experimentales para el caso de tener como cliente a la laptop HP Pavilion.

Las figuras 6.1 y 6.4 comparan el tiempo de ejecución de RSA predicho por las ecuaciones (1) a (4) contra el tiempo obtenido del experimento verdadero. Asimismo, las figuras 6.2 y 6.5 comparan el tiempo

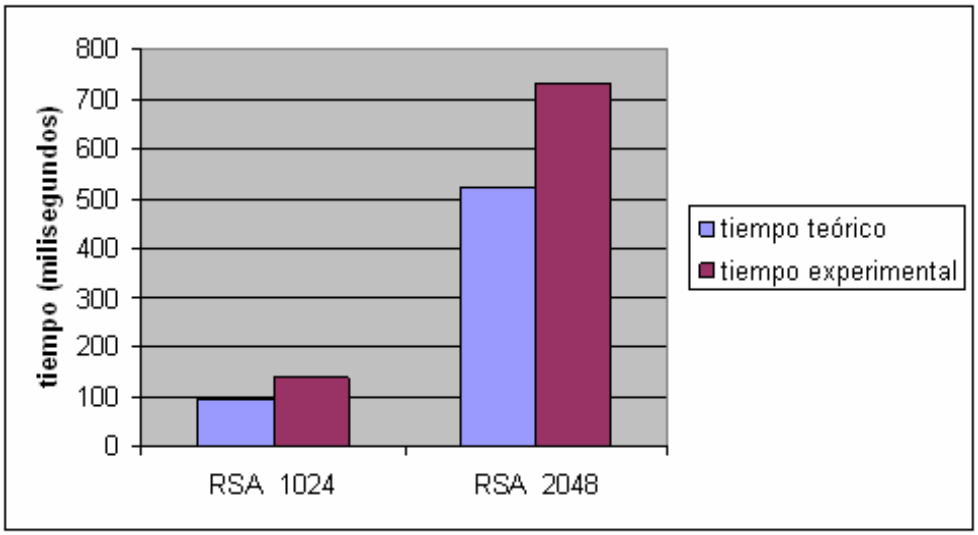

Figura 6.1: Tiempo teórico vs. tiempo experimental en RSA

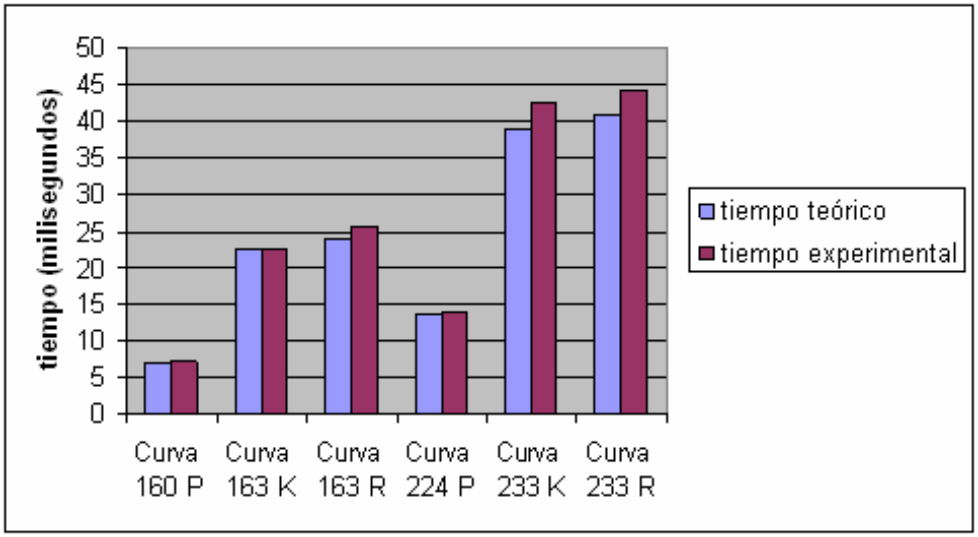

Figura 6.2: Tiempo teórico vs. tiempo experimental en ECC.

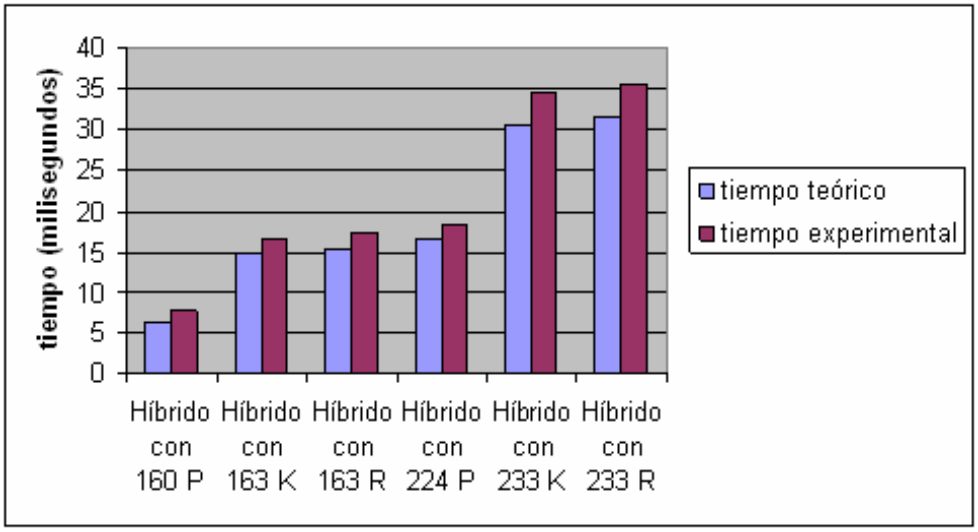

Figura 6.3: Tiempo teórico vs. tiempo experimental en esquema Híbrido

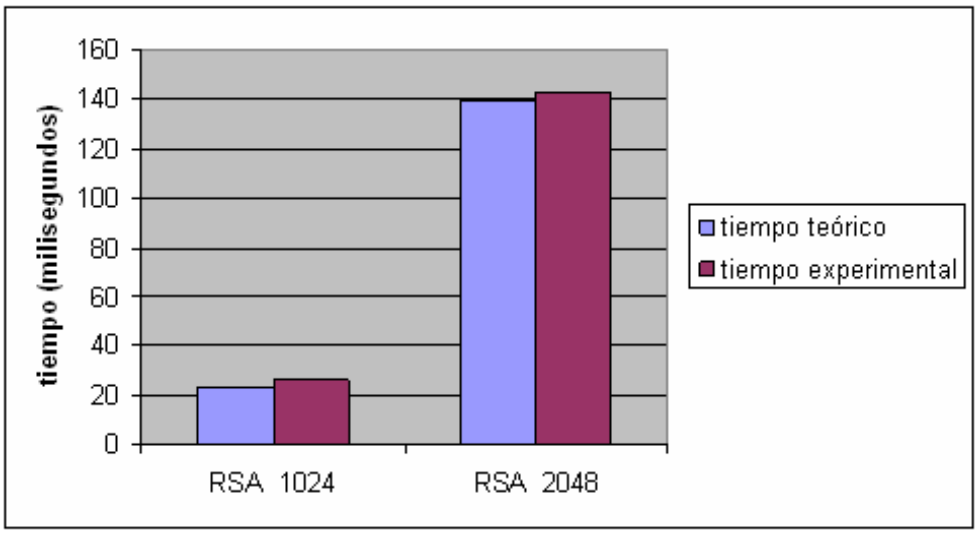

Figura 6.4: Tiempo teórico vs. tiempo experimental en RSA

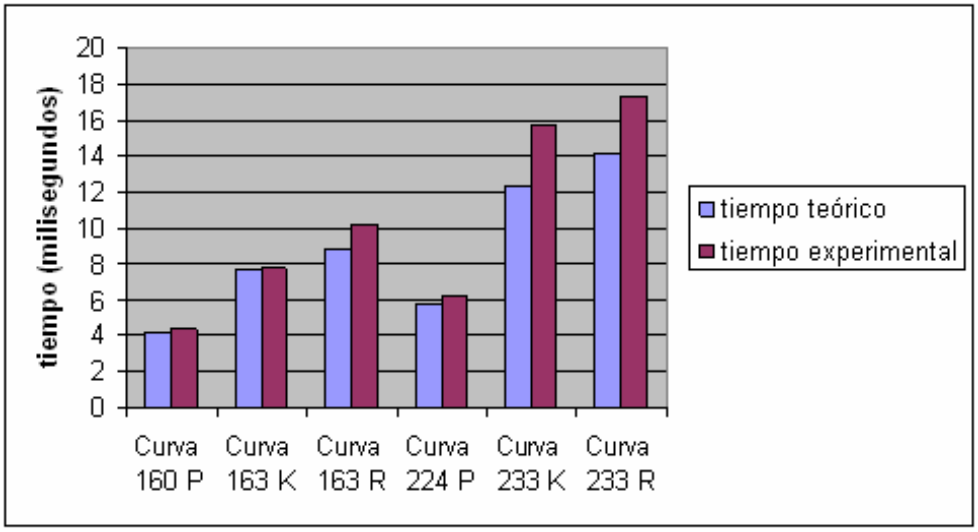

Figura 6.5: Tiempo teórico vs. tiempo experimental en ECC

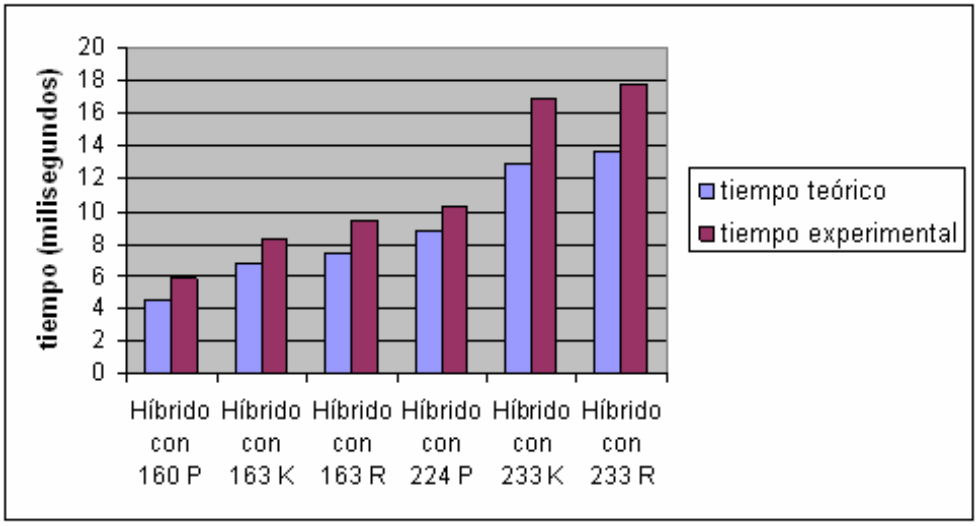

Figura 6.6: Tiempo teórico vs. tiempo experimental en esquema Híbrido

de ejecución de ECC predicho por las ecuaciones (1), (2), (5) y (6) contra el tiempo obtenido experimentalmente. Finalmente las figuras 6.3 y 6.6 comparan el tiempo de ejecución del esquema híbrido predicho por las ecuaciones (1), (2), (7) y (8) contra el tiempo obtenido experimentalmente.

Las discrepancias en las predicciones teóricas pueden deberse a que no se consideran diversas operaciones no criptográficas necesarias para la implementación del protocolo, tales como el análisis de los certificados del cliente y del servidor, obtención de los parámetros para RSA y para las curvas elípticas que están almacenados en archivos en memoria secundaria, entre otras.

Los resultados mostrados en las figuras anteriores son suficientes para concluir que ECC y el esquema híbrido superan a RSA en el protocolo de negociación de WTLS.

6.3. Análisis de resultados.

En este trabajo se realizó un prototipo del protocolo de negociación de WTLS que ha sido puesto en ejecución en un ambiente inalámbrico aún limitado debido a que los experimentos se han realizado con 2 dispositivos móviles típicos conectado con un servidor, siendo este una estación de trabajo. Los datos recopilados demuestran que ECC supera consistentemente la opción tradicional representada por RSA en todos los escenarios evaluados. Además el esquema híbrido propuesto que utiliza tanto operaciones criptográficas de ECC como de RSA, demuestra un buen desempeño al utilizar llaves pertenecientes al nivel de seguridad 1 mostrado en la Tabla 5.1, superando incluso a ECC. Sin embargo, para el nivel de seguridad 2 su desempeño con respecto a ECC se ve disminuido. Además, nuestras predicciones basadas en el modelo analítico demuestran una concordancia razonable con los datos experimentalmente obtenidos.

En las figuras 6.7 y 6.8 podemos observar el mejor y el peor caso con base al tiempo de la implementación del protocolo de negociación de WTLS clase 3. Como podemos observar, el mejor caso pertenece a ECC tanto en el nivel de seguridad 1 como en el 2 y también en ambas plataformas de prueba. Asimismo, el peor caso es para RSA. Considerando las restricciones de poder de cómputo y de memoria de los dispositivos móviles, RSA es la peor opción a elegir para proveer seguridad.

Los resultados obtenidos en este trabajo de tesis se publicaron en [25], [26]. En dichos trabajos se reportó, para el protocolo de Negociación de WTLS clase 3, diferencias de tiempos entre ECC y RSA hasta de 26.7 veces mayores, comparando ECC 160P y RSA 1024 para el nivel 1. Para el nivel 2 se halló una diferencia de 70 veces al comparar ECC 224P contra RSA de 2048. En este trabajo, al implementar el protocolo y ejecutarlo se encuentra, para el protocolo de Negociación de WTLS clase 3, diferencias de

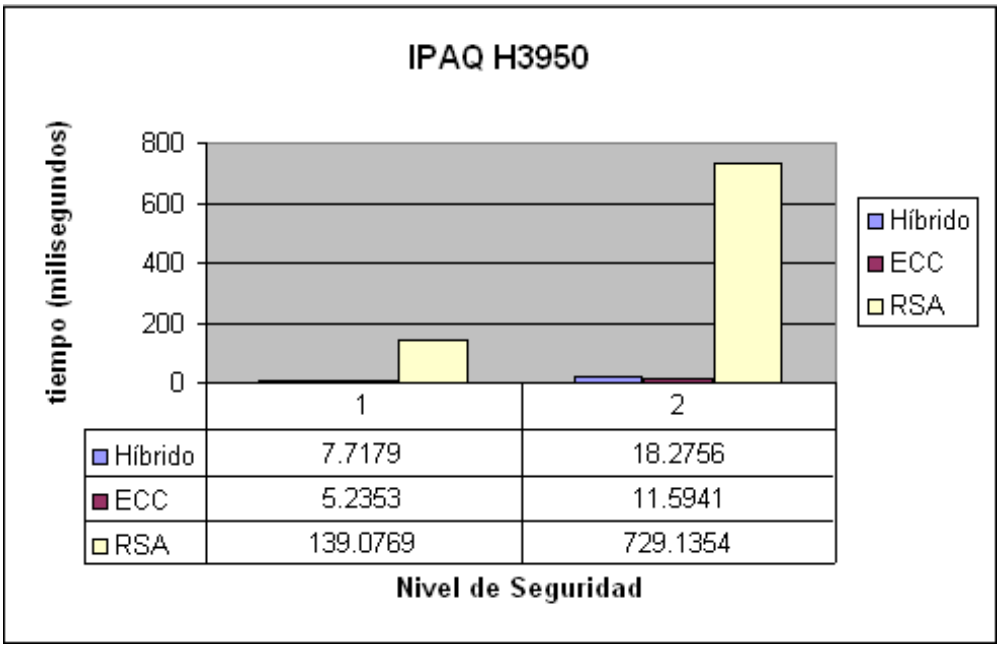

Figura 6.7: Comparativo entre ECC, esquema Híbrido y RSA para WTLS clase 3

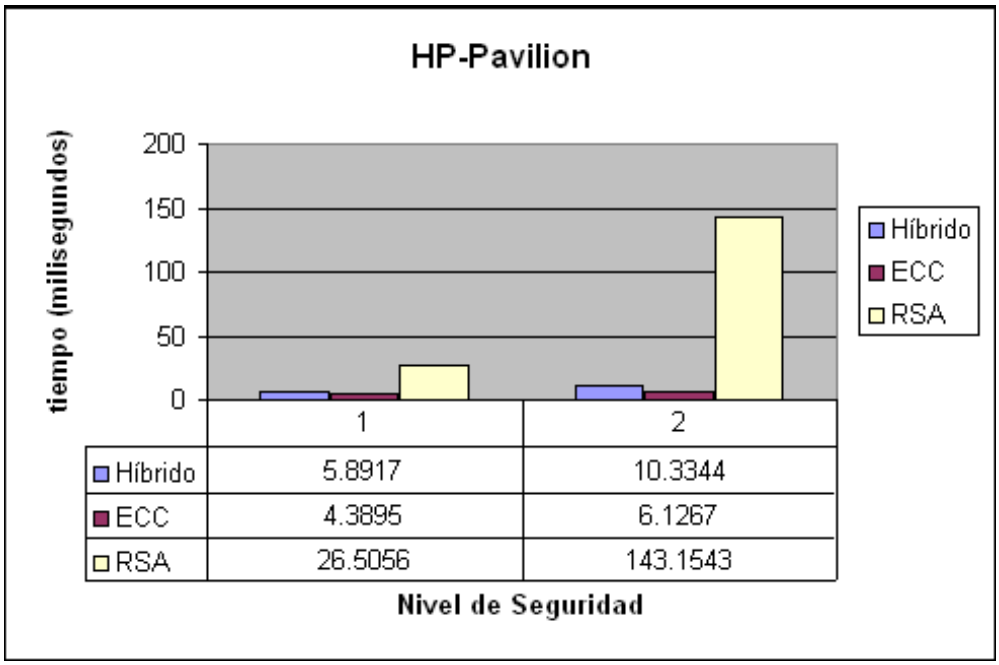

Figura 6.8: Comparativo entre ECC, esquema Híbrido y RSA para WTLS clase 3

tiempos entre el esquema Híbrido, ECC y RSA hasta de 18 veces y 26 veces mayores, comparando el esquema Híbrido usando la curva elíptica 160 P, ECC 160P contra RSA 1024 para el nivel 1. Para el nivel 2 se encontró una diferencia de aproximadamente 39 veces y 64 veces al comparar el esquema Híbrido usando la curva elíptica 224 P, ECC 224P contra RSA de 2048. Estos resultados son en el escenario en donde el cliente es la IPAQ H3950.

Para el caso de cuando el cliente es una laptop HP Pavilion, al implementar el protocolo y ejecutarlo se encuentra, para el protocolo de Negociación de WTLS clase 3, diferencias de tiempos entre el esquema Híbrido, ECC y RSA hasta de 4 veces y 6 veces mayores, comparando el esquema Híbrido usando la curva elíptica 160 P, ECC 160 P contra RSA 1024 para el nivel 1. Para el nivel 2 se encontró una diferencia de aproximadamente de 13 veces y 23 veces al comparar el esquema Híbrido usando la curva elíptica 224 P, ECC 224P contra RSA de 2048.

Dichos resultados demuestran que el criptosistema de curvas elípticas y el esquema Híbrido que se esta proponiendo son una mejor opción para proveer seguridad a dispositivos con restricciones computacionales.

Conclusiones

En este capítulo se presentan, de forma general, los resultados que se obtuvieron con el desarrollo de este trabajo de tesis. Asimismo, se describen las actividades que se pueden realizar para mejorar el mismo.

Con el desarrollo de este trabajo de tesis se obtuvo un sistema que es capaz de intercambiar datos de forma segura entre un dispositivo móvil ligero con un servidor, esto gracias a la utilización de las herramientas que brinda la criptografía. En este sistema, se implementó el protocolo de negociación de WTLS tratando de cumplir en gran parte la especificación WAP-261-WTLS-20010406-a Versión 6-Abril-2001 para crear una llave de sesión, con la opción de autenticar a las entidades, ya sea utilizando RSA, ECC o el modelo híbrido propuesto en este trabajo.

El modelo híbrido, que utiliza tanto operaciones criptográficas de ECC como de RSA, demostró un buen desempeño al utilizar llaves pertenecientes al nivel de seguridad 1, mostrado en la tabla 6.1, superando incluso a ECC. Sin embargo, para el nivel de seguridad 2 su desempeño con respecto a ECC se ve disminuido. Los datos recopilados demuestran que ECC supera consistentemente la opción tradicional representada por RSA en todos los escenarios evaluados.

Dado que uno de los principales objetivos de este proyecto fue brindar seguridad en la transferencia de información en un medio inalámbrico en el menor tiempo posible, se desarrolló un modelo analítico para estimar el tiempo que tomaría entablar una sesión segura. Las predicciones, basadas en dicho modelo, demuestran una concordancia razonable con los datos experimentalmente obtenidos.

Las contribuciones que se obtuvieron durante el desarrollo de esta tesis son las siguientes:

- Migración de la biblioteca criptográfica para que pudiera ser utilizada por un dispositivo móvil ligero.
- Migración del protocolo de negociación de WTLS para poder utilizarse en un dispositivo móvil ligero y en un ambiente inalámbrico.

- Implementación del modelo híbrido, el cual combina las ventajas de los algoritmos de llave pública RSA y ECC, para proveer una opción eficiente y segura de autenticación durante el protocolo de autenticación.
- Realización del modelo analítico del protocolo de negociación con el cual se puede estimar el tiempo de duración de dicho protocolo considerando el tiempo requerido por cada una de las operaciones criptográficas.
- Desarrollo de un sistema que permite intercambiar información de forma segura en un ambiente inalámbrico utilizando el protocolo de negociación de WTLS y criptografía de llave pública (RSA y ECC) para establecer una sesión segura. Además, emplea criptografía de llave simétrica para proveer el servicio de confidencialidad. Asimismo, el sistema ofrece los siguientes servicios de seguridad: integridad y autenticación utilizando firma/verificación digital y certificados digitales.

6.4. Trabajo futuro

Los siguientes puntos se pueden considerar para realizar mejoras a este trabajo de tesis.

- Dado que en este trabajo únicamente se hicieron pruebas con dos clientes es recomendable añadir más nodos y realizar pruebas de rendimiento, es decir, evaluar el efecto de que una mayor cantidad de nodos realicen peticiones al servidor y verificar el número máximo de éstos que el servidor puede atender sin que se pierda la calidad del servicio.
- Como los certificados utilizados en este trabajo se diseñaron de acuerdo a las necesidades del proyecto -siguiendo el estándar X.509-, es necesario analizar la posibilidad de que éstos se apeguen al estándar ASN.1 para que sean legibles desde cualquier plataforma.
- A lo largo del desarrollo del proyecto sólo se utilizo una red ad-Hoc. La pregunta que surge en este punto es: ¿Como se comportará el servidor si se utiliza una red con configuración de Infraestructura?
- Otra mejora que se puede hacer al sistema es añadir más algoritmos de cifrado simétrico.
- Refinar el modelo analítico para considerar el tiempo de ejecución de las operaciones no criptográficas y así obtener tiempos estimados más próximos a los reales.

Apéndice A

Funcionamiento del sistema

El sistema tiene una interfaz gráfica que muestra al usuario las opciones con las cuales se cuenta para realizar una sesión de intercambio seguro de datos. La pantalla principal sólo permite al usuario que inicie el protocolo de negociación para poder establecer una sesión segura. La figura A.1 muesta dicha pantalla.

La siguiente pantalla da al usuario la opción de elegir qué clase de implementación del protocolo de WTLS desea utilizar. En este caso se eligió la clase 3. La figura A.2 muestra la pantalla para elegir la clase de WTLS.

Al elegir la clase 3 se presentan las tres opciones de autenticación que soporta el protocolo, la opción que se va a elegir es la autenticación utilizando el modelo híbrido desarrollado en este trabajo. La figura A.3 muestra la pantalla con las opciones de autenticación.

Cuando se elige el modelo híbrido la información que se pide es el tamaño de las llaves para RSA, la figura A.4 muestra la pantalla donde se dan las opciones de los tamaños de llave soportados. Posteriormente se pide la curva elíptica que se utilizará para ECC. Dicha opción se muestra en la figura A.5.

Finalmente, se pide al usuario que elija el algoritmo de cifrado simétrico con el cual se podrán intercambiar datos de manera confidencial. La figura A.6 muestra la pantalla donde se muestran todos los cifradores simétricos disponibles.

Después de elegir todos los parámetros necesarios para entablar la sesión, el sistema inicia el protocolo de negociación y comienza la comunicación entre cliente y servidor. En la consola del servidor se muestran los parámetros que se acordaron y finalmente se muestra la llave de sesión generada. Para propósitos de demostración dicha llave de sesión se manda a consola. La salida en el servidor se muestra en la figura A.7.

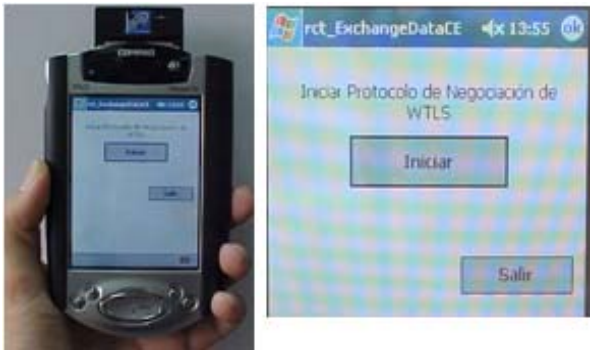

Figura A.1: Pantalla principal del sistema

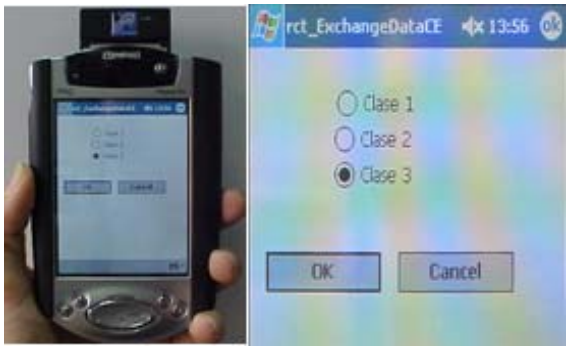

Figura A.2: Clases de WTLS

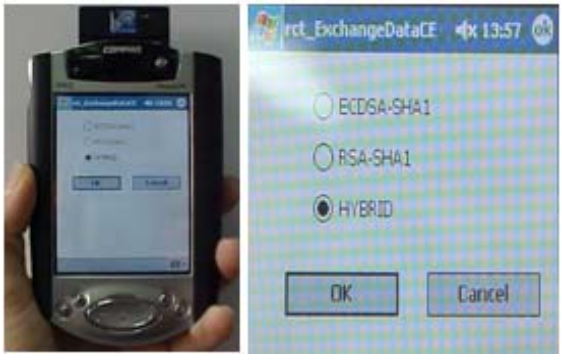

Figura A.3: Opciones de autenticación en clase 3

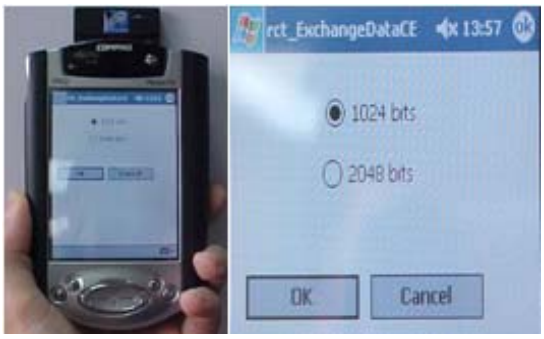

Figura A.4: Longitud de llaves en RSA.

Figura A.5: Curvas disponibles para ECC

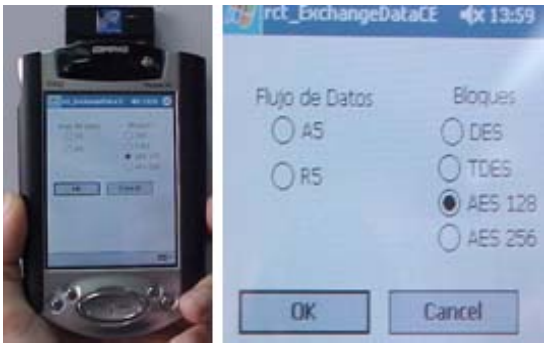

Figura A.6: Cifradores simétricos

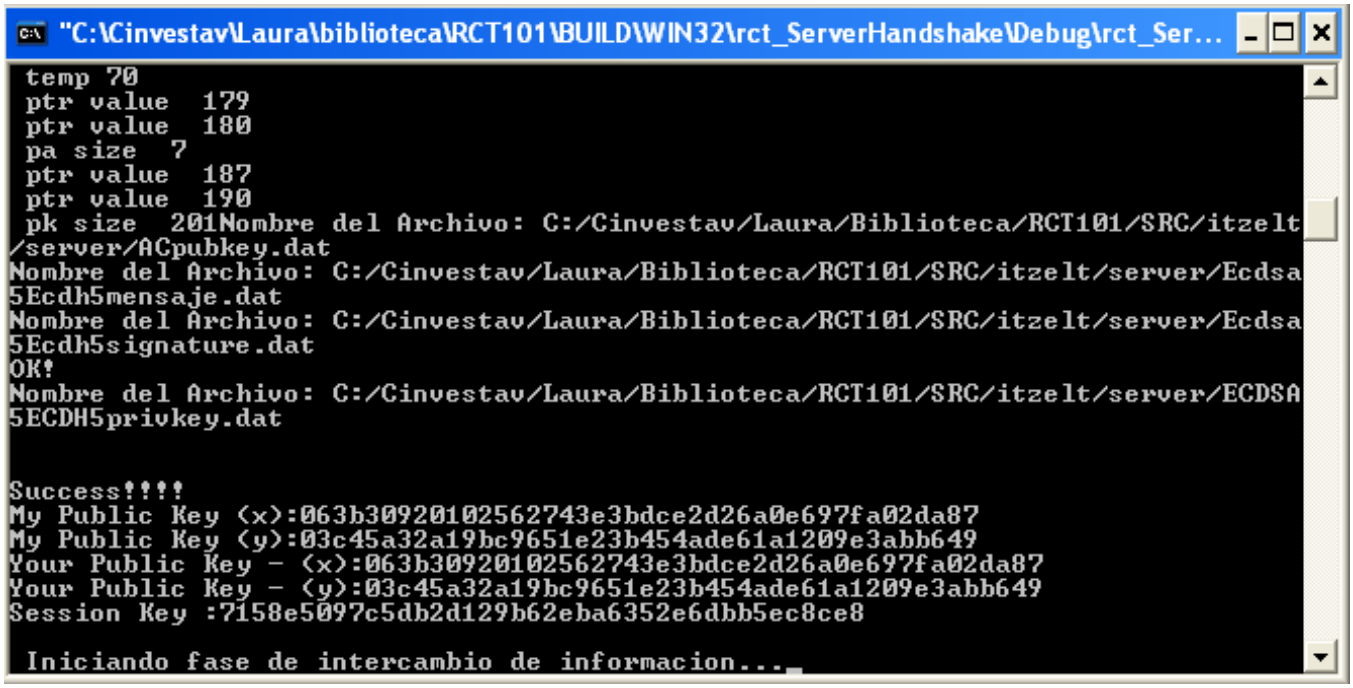

Figura A.7: Salida en el servidor al terminar protocolo de negociación

Figura A.8: Opciones de intercambio de datos

La figura A.8 muestra las opciones de intercambio de datos que se presentan al usuario. Como se puede observar, están el cifrar, descifrar, firmar y verificar datos.

Si el usuario elige enviar un archivo cifrado o firmado digitalmente al servidor, primero debe escoger el archivo que va a enviar. Para ello, se despliega la pantalla mostrada en la figura A.9. Cuando termina el proceso de cifrado se despliega un aviso el cual se muestra en la figura A.10. Para el caso en que el usuario elija firmar un documento, el sistema pide una contraseña para recuperar la llave privada del usuario, la cual está almacenada de forma cifrada. La contraseña se hace pasar por la función hash MD5 para crear una llave de 128 bits que se usará para descifrar la llave privada. El algoritmo usado para descifrar esta llave es AES. Mientras que el usuario no escriba la contraseña correcta, el sistema no lo dejará realizar más operaciones. Desde luego, el usuario siempre tiene la opción de cancelarla. La pantalla que pide la contraseña del usuario se muestra en la figura A.11. Si el proceso de firmado digital es exitoso el sistema avisa al usuario con la pantalla mostrada en la figura A.12.

En el caso de que el usuario elija las opciones de descifrar o verificar, el sistema le mostrará los archivos que el servidor tiene disponibles para que escoja uno de ellos y le sea enviado. La figura A.13 muestra la pantalla donde se despliegan los archivos que el servidor tiene disponibles para ser intercambiados. Cuando el proceso culmina exitosamente se despliegan pantallas de aviso como las mostradas anteriormente.

En el momento en que el usuario decida terminar la sesión sólo es necesario que oprima el botón de salir para finalizar la conexión inalámbrica y dar por terminado el intercambio de datos.

Figura A.9: Elección de archivo

Figura A.10: Terminación del proceso de cifrado

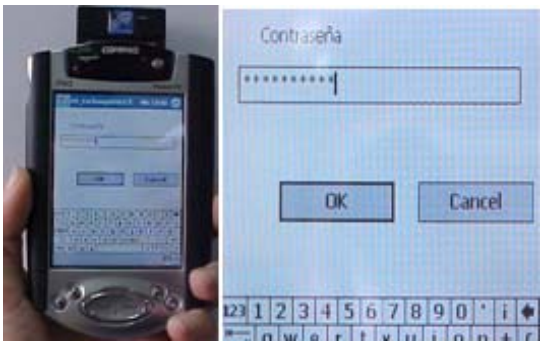

Figura A.11: Contraseña del usuario

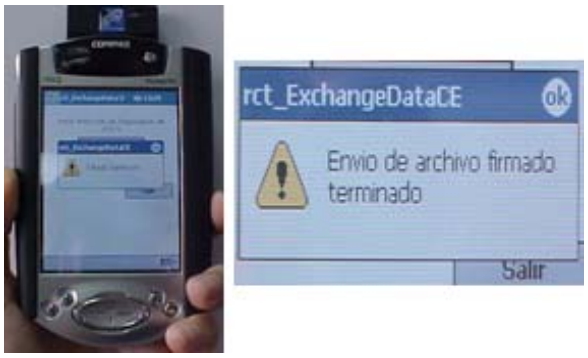

Figura A.12: Terminación de proceso de firmado digital exitoso.

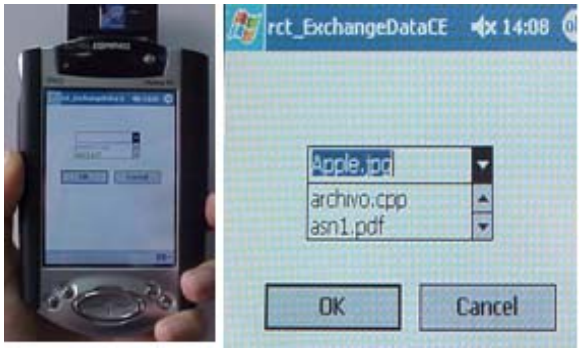

Figura A.13: Lista de archivos disponibles en el servidor.

Apéndice B

Wireless Application Protocol (WAP)

El Wireless Application Protocol (WAP) es el resultado del trabajo continuo para definir una especificación para el desarrollo de aplicaciones que operen sobre redes de comunicación inalámbrica. El campo de acción del foro WAP es definir un conjunto de especificaciones que deben ser usadas por servicios de aplicación. El mercado inalámbrico está creciendo rápidamente, buscando nuevos clientes y ofreciendo nuevos servicios. WAP selecciona y define un conjunto de protocolos extensibles y abiertos, como base para implementaciones interoperables.

Los objetivos del Foro WAP [33] son:

- Traer el contenido de Internet y las ventajas de los servicios de datos a los teléfonos celulares digitales y otras terminales inalámbricas (PDAs, por ejemplo).
- Crear una especificación de un protocolo inalámbrico global que trabaje a través de diferentes tecnologías inalámbricas.
- Habilitar la creación de contenidos y aplicaciones escalables en un rango amplio de tipos de dispositivos.
- Abarcar y extender los estándares y la tecnología existente, siempre que sea apropiada.

WAP consta de varios niveles que contienen la mayoría de las características del modelo de referencia OSI. WAP está dividido en 5 niveles:

1. Capa de aplicación (WAE – Wireless Application Environment). Sirve para el desarrollo de aplicaciones portátiles usando Wireless Markup Language (WML) y WMLScript.

2. Capa se sesión (WSP – Wireless Session Protocol). Define el formato de los mensajes al momento de ser transmitidos. Cabe destacar que WSP es la versión binaria del protocolo HTTP.

3. Capa de transacción (WTP – Wireless Transaction Protocol). Este protocolo es el encargado del control de los mensajes.

4. Capa de seguridad (WTLS – Wireless Transport Layer Security). Provee una interfaz de servicio de transporte segura que preserva a la interfaz de servicio de transporte

5. Capa de transporte (WDP – Wireless Datagram Protocol). Este protocolo permite la transmisión de mensajes sin conexión, esto es, tiene mecanismos para segmentar y ensamblar los paquetes. Este protocolo es autoadaptable a las características de la red inalámbrica en la cual está trabajando.

La estructura de WAP se muestra en la figura B.1.

Capa de Seguridad

La capa WTLS está compuesta de dos niveles lógicos como se muestra en la figura B.2.

En el nivel inferior se encuentra el protocolo de Registro. En este nivel se lleva a cabo la fragmentación de los mensajes, así como el cifrado de los mismos utilizando cifradores por bloques. En este nivel se lleva a cabo el firmado digital para garantizar la integridad de los datos.

En el nivel superior se encuentra un conjunto de protocolos: el protocolo de Alerta, el protocolo de Aplicación, el protocolo de Negociación y el protocolo de Cambio de Cifrado.

- El protocolo de Alerta tiene como actividades emitir avisos cuando se presentan problemas en la fase de gestión de una sesión de seguridad. Este protocolo contempla 3 tipos de avisos dependiendo de la gravedad del evento: fatal, crítico y de advertencia.

- El protocolo de Aplicación es la interfaz para las capas superiores.

- El protocolo de Cambio de Cifrado se encarga de transmitir los datos cifrados utilizando los algoritmos de ciframiento acordados en la fase de negociación.

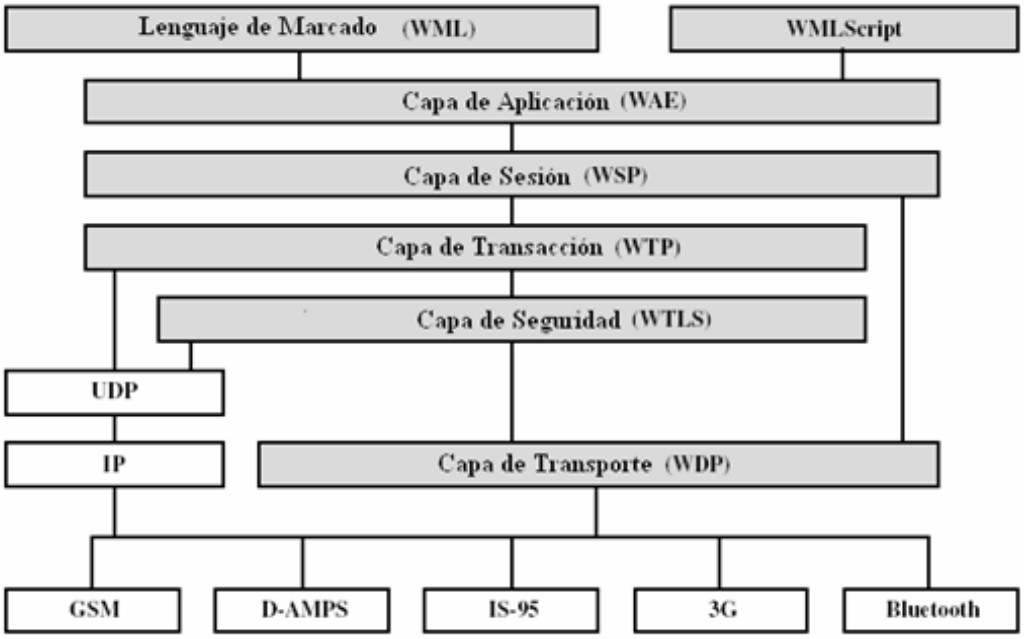

Figura B.1: Pila de protocolos de WAP

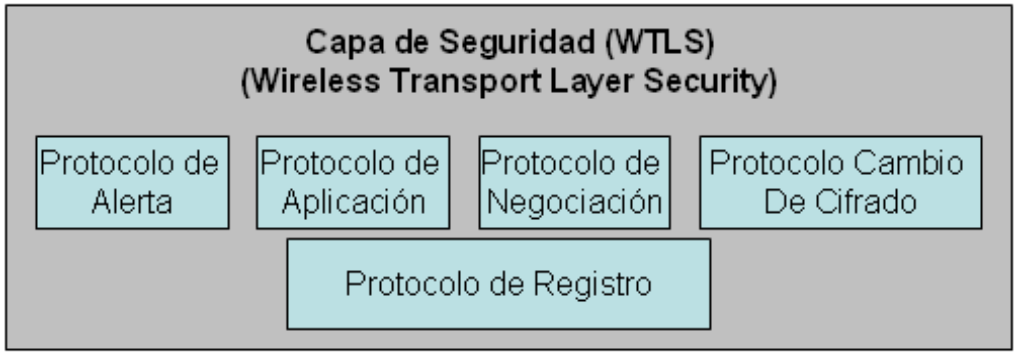

Figura B.2: Niveles de WTLS

- En el protocolo de Negociación el cliente y el servidor establecen acuerdos acerca de los protocolos que van a utilizar, se establecen los algoritmos criptográficos para el ciframiento de la información, se realizan las autenticaciones mutuas y se intercambian la llave de sesión que se utilizará para cifrar y descifrar mientras la sesión se mantenga.

Apéndice C

Características del sistema

El Sistema de Seguridad para Intercambio de Dispositivos Móviles es un sistema que permite al usuario intercambiar información de manera confidencial o firmada electronicamente de una manera simple en ambientes inalámbricos. El sistema cuenta con una serie de servicios y herramientas con las cuales se provee un marco de seguridad para el intercambio de datos tanto en el servidor como en dispositivos móviles que tengan una plataforma Windows Móbile.

Además:

- Se establece una sesión segura con base al protocolo de negociación de WTLS.
- Envía documentos de forma confidencial y segura.
- La autenticación de usuarios se realiza por medio de certificados digitales.

A continuación se muestran los requerimientos mínimos para utilizar el Sistema para Intercambio de Datos en Dispositivos Móviles

Especificaciones Técnicas Servidor

- El sistema requiere sistema operativo Microsoft® 32 bits preferible a partir de Windows 98 o superior.
- Tarjeta de red inalámbrica configurada como ad-hoc y habilitada

Especificaciones Técnicas Cliente (usuario final)

- Microsoft® Windows Mobile.
- Tarjeta de red inalámbrica configurada como ad-hoc y habilitada.
- Tener como puerta de enlace la IP de la máquina servidor.

De manera general la tabla 7.1 describe las características más importantes del sistema.

Servicios de Seguridad	Autenticación Integridad Confidencialidad No-Repudio
Algoritmos Simétricos	Por Flujo de Datos: A5, RC4 Por Bloques: DES, Triple DES, AES
Algoritmos Asimétricos	RSA Criptografía de Curvas Elípticas Híbrido (RSA y ECC)
Funciones Hash	MD5 SHA-1
Algoritmos para Firma/Verificación Digital	ECDSA PKCS
Protocolos de Seguridad	Diffie-Hellman Elliptic Curve Diffie-Hellman (ECDH) TLS/WTLS
Certificados Digitales	Apegados al estándar X.509
Tipos de Autenticación	Anónima Autenticación solo del Servidor (usando certificados) Autenticación de Cliente y Servidor (usando certificados)

Tabla 7.1 Características del Sistema

Bibliografía

[1] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, New York, quinta edición, 2001.

[2] B. Schneier, *Applied Cryptography. Protocols, Algorithms, and Source Code in C*. Wiley, USA, segunda edición, 1996.

[3] W. Diffie and M. Hellman, New directions in cryptography. *IEEE Transaction on Information Theory*, IT-22(6), pp. 644–654, 1976.

[4] ETSI HIPERLAN/2 Standar. Disponible en: http://portal.etsi.org/bran/kta/Hiperlan/hiperlan2.asp

[5] Intel Centrino Mobile Technology. Disponible en: http://www.intel.com/products/mobiletechnology/

[6] M. Maxim, D. Pollino, *Wireless Security*. McGraw Hill. Berkeley, California. 2002.

[7] L. Lucena, *Criptografía y Seguridad en Computadores*. Tercera Edición, Versión 1.00, 2001. Libro electrónico. Disponible en: http://wwwdi.ujaen.es/~mlucena/

[8] Institute of Electrical and Electronics Engineers. Disponible en: http://www.ieee.org

[9] Grupo de trabajo del estándar IEEE 802.11i. Disponible en: http://grouper.ieee.org/groups/802/11/

[10] S. Petrovic, A. Fúster. Criptoanálisis del algoritmo A5/2 para telefonía móvil. *Primer Congreso Iberoamericano de Seguridad Informática* CIBSI'02, Morelia, México. 2002

[11] A. Biryukov, A. Shamir, D. Wagner. Real Time Cryptanalysis of A5/1 on a PC, *Lecture Notes In Computer Science*; Vol. 1978, *Proceedings of Fast Software Encryption Workshop* 2000, April 10-12, 2000, New York City.

[12] Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197 (FIPS PUB197). November 26, 2001. Disponible en: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[13] Data Encryption Standar (DES). Federal Information Processing Standards Publication 46-3 (FIPS PUB 46-3). October 25, 1999. Disponible en: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[14] L. C. Washington, *Introduction to Cryptography with Coding Theory*. Prentice Hall, New Jersey, 2002.

[15] W. Stalling, *Cryptography and Network Security, Principles and Practice*. Prentice Hall, Third edition, New Jersey 2003.

[16] D. Johnson, A. Menezes, S. Vanstone, The Elliptic Curve Digital Signature Algorithm (ECDSA). *International Journal of Information Security*, Vol. 1, Num. 1, August 2001.

[17] PKCS #1 v2.1: RSA Cryptography Standard. RSA Laboratories. Junio, 2002. Disponible en: http://www.rsasecurity.com/rsalabs/.

[18] PKCS #7: Cryptographic Message Syntax Standard. RSA Laboratories. November 1, 1993. Disponible en: http://www.rsasecurity.com/rsalabs/.

[19] WAP-261-WTLS-20010406-a, Wireless Transport Layer Security, Version 6-Abril-2001. Disponible en: http://www.wapforum.com

[20] R. Pressman, *Ingeniería del Software, un Enfoque Práctico*. McGraw-Hill, 1993. Tercera Edición.

[21] E. Savas, F. Rodríguez-Henríquez, C. Koc, et al. RCT, RSA and ECC Toolkit. ISL Group, Oregon State University. Marzo 2002.

[22] A. Levi, E. Savas. Performance Evaluation of Public-Key Cryptosystem Operations in WTLS Protocol. *The 8th IEEE Symposium on Computers and Communications* – ISCC 2003, Kemer-Antalya, Turkey, June 30 – July 3, 2003.

[23] L. Reyes-Montiel y F. Rodríguez-Henríquez, Estudio Comparativo de los Sistemas Criptográficos de Clave Pública de WTLS, *2do Congreso Iberoamericano de Seguridad informática* CIBSI 2003, pp. 356-366, Octubre 28-31 2003, Ciudad de México.

[24] V. Gupta, S. Gupta, S. Chang and D. Stebila, *Performance Analysis of Elliptic Curve Cryptography for SSL*, Proc. ACM Workshop on Wireless Security 2002, p.p. 87-94, ACM Press, September 2002, Atlanta.

[25] C. E. López-Peza, F. Rodríguez-Henríquez, M. A. León-Chávez, Comparative Performance Analysis of Public-Key Cryptographic Operations in the WTLS Handshake Protocol. *1^{st}. International Conference on Electrical and Electronics Engineering (ICEEE)*, Septiembre 8-10 2004, Acapulco, Guerrero, México.

[26] C. E. López-Peza, F. J. Rodríguez-Henríquez, M. A. León-Chávez. Análisis de desempeño del protocolo de negociación de WTLS en un ambiente inalámbrico multinodo. *2° Congreso Internacional en Innovación y Desarrollo Tecnológico* CIINDET 04, Noviembre 17-19 2004, Cuernavaca, Morelos, México.

[27] N. Borisov, I. Goldberg, D. Wagner, "Intercepting mobile communications: The insecurity of 802.11", en *Proceedings of the Seventh Annual International Conference on Mobile Computing and Networking*, Association for Computing Machinery, New York, NY, p. 180, 2001.

[28] I. Herwono, I. Liebhardt, Performance Evaluation of the WAP Security Protocols, *Proceedings of the 10th Aachen Symposium on Signal Theory*, Sept. 2001, pp. 95-100, Aachen, Germany. Disponible en: http://www.comnets.rwth-aachen.de

[29] I. Herwono, I. Liebhardt Performance of WTLS and its Impact on an M-Commerce, *Proceedings of the Third International Conference on Information and Communications Security*, Nov. 2001, pp. 167 - 171, Xi'an, China. Disponible en: http://www.comnets.rwth-aachen.de.

[30] M. Brown, D. Cheung, D. Hankerson, J. Hernandez, M. Kirkup, and A. Menezes. PGP in Constrained Wireless Devices, en *9th USENIX Security Symposium*, p.p. 247-261, Agosto 2000.

[31] D. Boneh, H. Shacham, and E. Rescrola. Client side caching for TLS, en *The Internet Society's 2002 Symposium on Network and Distributed System Security* (NDSS), pp. 195–202. Disponible en: http://crypto.stanford.edu/~dabo/pubs.html

[32] L. Reyes-Montiel, Estudio, diseño y evaluación de protocolos de autenticación para redes Inalámbricas, Tesis de Maestría presentada en la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de Investigación y de Estudios Avanzados del IPN, México, D. F., Noviembre de 2003.

[33] WAP Forum. WAP white paper. Disponible en http://www.wapforum.com.

[34] X. Wang, Y. Lisa Yin and H. Yu. Collision Search Attacks on SHA-1. Disponible en: http://theory.csail.mit.edu/~yiqun/shanote.pdf

[35] Certicom Securing Innovation. Disponible en: http://www.certicom.com/index.php

[36] Cryptovision: advanced cryptographic technology. Disponible en: http://www.cryptovision.com/cvweb/index.php