UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
COLEGIO DE CIENCIAS Y HUMANIDADES
UNIDAD ACEDÉMICA DE LOS CICLOS PROFESIONAL Y DE POSGRADO

PROTOCOLOS CRIPTOGRÁFICOS AUTENTICACIÓN E INTERCAMBIO DE LLAVES BASADOS EN PASSWORDS

TESIS

QUE PARA OBTENER EL TÍTULO DE:

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

HERNÁNDEZ AUDELO, LEOBARDO

Ciudad Universitaria, Distrito Federal, 1999

1999

RESUMEN

En este trabajo se hace un estudio de la problemática de seguridad inherente a los sistemas que utilizan protocolos criptográficos basados en secretos cortos y memorazibles llamados passwords para lograr autenticación e intercambio de llaves. Se analizan los ataques a los que son vulnerables estos protocolos, las principales alternativas y soluciones existentes, y el estado del arte en las principales direcciones de investigación en esta área.

Actualmente se pueden distinguir dos principales corrientes de investigación enfocadas a la búsqueda de soluciones al difícil problema de lograr autenticación e intercambio o acuerdo de llave de modo seguro, basándose solamente en passwords débiles. Una de estas vertientes utiliza alguna técnica criptográfica para cifrar los mensajes intercambiados entre las partes involucradas en el protocolo; la otra utiliza relaciones matemáticas que no requieren cifrar esos intercambios. A los protocolos de estas dos vertientes se les denomina fuertes en el sentido de que, aún cuando asumen el uso de passwords débiles, logran autenticación e intercambio de llaves de manera segura.

En este trabajo se hace énfasis en el estudio y análisis de las soluciones criptográficas fuertes debido a que, en la práctica, son las más viables en términos de requerimientos, sencillez, y eficiencia para lograr implementaciones concretas.

A partir del estudio y análisis de las diferentes soluciones criptográficas fuertes que existen, se hace un análisis comparativo de todas ellas en función de sus características, suposiciones, propiedades, ataques que soportan, ventajas y desventajas. Este estudio es el primero en su género de que se tiene conocimiento a la fecha.

En este trabajo también se construye una estructura arborescente que muestra el desarrollo y evolución en el tiempo de estos protocolos, sus influencias y derivaciones hasta el momento. Un trabajo similar a esta clasificación se presenta en [52] pero no incluye todos los protocolos presentados en este trabajo y no menciona influencias específicas.

Igualmente, se presentan los resultados obtenidos de la búsqueda de infor-

mación y referencias acerca de implementaciones concretas y sobre pruebas y estudios de eficiencia de este tipo de protocolos.

Finalmente, se mencionan los distintos ámbitos y aplicaciones en los que este tipo de protocolos pueden ser implementados, mencionando sus ventajas con respecto a las alternativas en uso actual, y la conveniencia de sustituir esas implementaciones actuales por soluciones fuertes.

Contenido

Capítulo 1

Introducción

En este capítulo se introducen los principales conceptos que se utilizan en este trabajo, empezando por los conceptos de comunicación segura y seguridad en cómputo; el importante concepto de autenticación y su integración con el intercambio de llaves en un mismo proceso; las distintas técnicas de autenticación que existen; la autenticación basada en passwords y sus distintos esquemas y mejoras posibles; también se presentan los conceptos de autenticación fuerte y protocolo criptográfico de autenticación. Finalmente, se muestra un bosquejo de la estructura general de este trabajo.

1.1 Comunicación Segura y Seguridad en Cómputo

En la actualidad, es ampliamente conocida la importancia de las comunicaciones seguras en ambientes distribuidos y de red [9](pp. 1). Se entenderá, en este contexto, como *seguridad en las comunicaciones* la acción de proteger datos que representan y codifican información durante su transmisión en redes de computadoras y sistemas distribuidos. El término *información* se refiere, de acuerdo a la Enciclopedia Webster, en general, al conocimiento comunicado o recibido concerniente a un hecho particular o circunstancia; y en las ciencias de la computación, se refiere a los datos que pueden ser codificados para su proceso por una computadora o dispositivo similar. El nombre genérico de *seguridad en cómputo* se asocia con la preservación de los

recursos de cómputo contra abusos y uso no autorizado, así como también con la protección de datos contra daño accidental o deliberado, divulgación y modificación [47] (pp. 1 - 2).

Al hablar de sistemas distribuidos y de red, es necesario definir de manera precisa lo que significan estos conceptos. De acuerdo a (Tanenbaum, 1988), el término *red de computadoras* se refiere a una colección interconectada de computadoras autónomas. Dos computadoras están interconectadas si son capaces de intercambiar datos. Son autónomas si no existe una relación maestro/esclavo entre ellas. Según Lamport, un *sistema distribuido* consiste de una colección de procesos distintos espacialmente separados, que se comunican unos con otros por intercambio de mensajes. Además, un sistema es distribuido si el retardo en la transmisión de mensajes no es despreciable comparado con el tiempo entre eventos en un único proceso [48].

1.1.1 Servicios de Seguridad

La importancia de la seguridad en las comunicaciones resulta evidente cuando una gran cantidad de servicios que requiere la sociedad moderna se solicitan y ofrecen por medio de sistemas en red. Tales servicios van desde el acceso remoto a sistemas de cómputo, cajeros automáticos, servicios bancarios telefónicos, etc., hasta compras por *Internet* [12]. Por esta razón, se han invertido muchos esfuerzos en proporcionar servicios de seguridad en gran variedad de ambientes de red y sistemas operativos.

Se conoce como *servicio de seguridad* a una funcionalidad específica que forma parte de la seguridad de un sistema y de su transferencia de información. Tal servicio contrarresta ataques a la seguridad y utiliza uno o más mecanismos para proporcionar el servicio. Se entiende por *mecanismo de seguridad* el mecanismo diseñado para detectar, prevenir o recobrarse de un ataque de seguridad. Un *ataque de seguridad* es la acción que compromete la seguridad de la información [47] (pp. 1 - 2). En la sección 2.3 de este trabajo se trata con más detalle el concepto de ataque.

La arquitectura de seguridad *OSI* (Estándares ISO 7498-2 y ITU-T X.800) distingue cinco clases de servicios de seguridad: *autenticación*, *control de acceso*, *confidencialidad* de los datos, *integridad* de los datos, y *no repudio* [47] (pp. 7 - 10).

El servicio de autenticación involucra un requerimiento de identidad o identificación corroborable y distingue entre la autenticación de las partes que se comunican entre sí y la autenticación del origen de los datos. Debido al objetivo de este trabajo, en la sección siguiente se revisará con detalle el tipo de autenticación que relaciona a las partes involucradas en una comunicación.

El control de acceso es un servicio que proporciona protección a los recursos del sistema contra acceso y uso no autorizado. Este servicio está íntimamente relacionado al de autenticación debido a que un usuario o proceso debe ser autenticado antes de tener acceso a cualquier recurso del sistema.

La confidencialidad, también conocida como privacidad o privacía, proporciona a los datos protección contra lectura o divulgación no autorizada. Este servicio oculta o transforma los datos de tal manera que sólo las partes autorizadas puedan enterarse de su contenido.

El servicio de integridad protege a los datos contra modificaciones, alteraciones, borrado, inserciones, y de todo tipo de acción que atente contra su integridad.

El no repudio proporciona protección contra la posibilidad de que alguna de las partes involucradas en una comunicación nieguen ya sea haber enviado un mensaje u originado una acción, o bien nieguen haber recibido un mensaje o haber sido el destinatario de cierta acción.

1.2 Autenticación

Uno de los servicios de seguridad que es básico para toda comunicación segura, es el que tiene que ver con la identificación de las partes que se comunican para solicitar o proporcionar un servicio. La parte que solicita el servicio debe convencer a la otra parte de su identidad; es decir, debe demostrar, de alguna manera, que es quien dice ser. Este proceso es el que se conoce como *autenticación*. La autenticación puede involucrar a dos o más partes; puede ser unilateral o mutua; las partes pueden compartir un secreto (caso simétrico) o no (caso asimétrico) [24] (pp. 4).

La autenticación es unilateral, si basta que una parte compruebe que la otra es quien dice ser. Si la autenticación se realiza sólo entre las partes directamente involucradas, sin que nadie más participe, se dice que se tiene *autenticación directa* [11] (pp. 1).

Con frecuencia, también se requiere que la parte que ofrece el servicio, a su vez, demuestre al solicitante que él es el otorgante legítimo del servicio; es decir, debe convencer a la otra parte que él es quien dice ser. Esto es lo que se conoce como *autenticación mutua*. Se llama *autenticación mutua de dos partes*, si, en el mismo proceso, las partes involucradas deben autenticarse una a la otra y viceversa.

Otra forma de autenticación es por vía indirecta. La parte que necesita identificarse ante otra, lo hace por medio de una tercera parte en la cual confían ambas. En este caso se conoce al protocolo de autenticación como de tres partes. La tercera parte actúa como *autoridad certificadora o juez*, quien asume la responsabilidad de asegurar a la parte que exige la identificación que el identificante es quien dice ser. En ámbitos de redes de computadoras y sistemas distribuidos, esta tercera parte es la que se conoce como *autoridad certificadora, servidor de llaves o centro de distribución de llaves.*

1.2.1 Autenticación e Intercambio de Llaves

Después que el proceso de autenticación se ha efectuado, lo que normalmente sigue en un ambiente de red es que las partes intercambien información relacionada al servicio mismo o al propósito de la comunicación. Este intercambio idealmente también debe ser hecho de manera segura y confidencial. Para lograr esto, una de las formas más ampliamente usadas en la actualidad consiste en intercambiar de manera segura, después de la autenticación, una llave o llaves para cifrar toda la información posteriormente intercambiada. Esta parte se conoce como *intercambio de llave* de sesión, porque normalmente la llave o llaves que se intercambian son únicas y útiles sólo para cada sesión.

Los procesos de autenticación e intercambio de llaves, normalmente se realizan uno enseguida del otro, razón por la cual la autenticación casi siempre es un preludio al intercambio de llaves y, a veces, al hablar de autenticación también se hace referencia implícita al intercambio de llaves [10] (pp. 1).

Debido a que el presente trabajo analiza procesos de autenticación basados en passwords, es necesario explicar con cierto detalle lo que esto significa. A continuación mencionamos las categorías que existen en técnicas de autenticación, para ubicar este trabajo en su contexto preciso.

1.2.2 Técnicas de Autenticación

Es común en la literatura dividir las técnicas de autenticación en tres categorías principales [38, 45, 47, 49]:

1. *Por conocimiento.-* Las partes directamente involucradas comparten el conocimiento sobre un secreto previamente acordado. Si se trata de autenticación indirecta, las partes comparten ese secreto con la autoridad certificadora. Este secreto compartido es lo que comúnmente se conoce como *password*, contraseña, palabra clave, etc. En este trabajo utilizaremos la palabra password como sinónimo de todas las demás.

2. *Por posesión.-* La parte que se identifica posee algo que debe mostrar a la otra parte, la que identifica, para convercerla de que es quien dice ser. Este algo puede ser una llave física o una tarjeta de identificación, tal como una licencia, credencial o pasaporte. Muchas veces, en esta categoría está implícita la autenticación indirecta, puesto que el objeto que se muestra para identificarse está avalado por una cierta autoridad certificadora tal como una institución pública o privada que es la que extiende la identificación, certificando con ello la identidad de quien se identifica.

3. *Por lo que se es.-* En esta categoría, la parte que se identifica muestra como prueba de identidad alguna característica única inherente a su propia naturaleza, tal como sus huellas digitales, imagen facial, o patrón de voz. La parte identificadora debe ser capaz de comparar esas características de la parte identificada, con las características originales que debe mantener almacenadas en algún lado.

1.3 Autenticación Basada en Passwords

A pesar de existir tres categorías, en los ambientes de redes y sistemas distribuidos, la mayoría de los mecanismos de autenticación en uso actualmente, se basan en prueba de identidad por conocimiento. Esto se debe a que esta prueba no requiere más que un password compartido, memorizado o almacenado de alguna forma en algún lugar; es implementable con relativa facilidad en sistemas de cómputo y redes; y no requiere dispositivos especiales y objetos físicos, tales como reconocedores de patrones y tarjetas de identificación. Al no tener estos requerimientos, su implementación tiene la ventaja adicional de ser eficiente y barata, comparada con las otras dos categorías [19](pp. 3).

A pesar de sus ventajas, las técnicas de autenticación por conocimiento presentan serias dificultades en la práctica: no es fácil pensar en un esquema de este tipo que permita realizar la autenticación sin exponer o revelar el password compartido ni siquiera a las partes involucradas; tampoco debe comprometerse ese password ya que debe ser almacenado de alguna manera por al menos una de las partes y, además, el password debe viajar por canales de comunicación inseguros, para ser verificado. En su viaje, debe estar asegurado de tal manera que sólo sea posible que lo conozcan las partes involucradas.

La problématica que surge alrededor de la búsqueda de esquemas seguros de autenticación basada en passwords es muy amplia y tiene que ver con el hecho de que ese password, que las partes involucradas comparten, generalmente, para ser práctico y memorizable, lo escoge un usuario, es decir, una persona, y esto hace que ese password sea débil en términos criptográficos [1], y por tanto fácilmente vulnerable. Además, en los sistemas tradicionales se guarda el password, de alguna forma y de cierto modo, en la parte que autentica. Al ser así, está expuesto a robos o lecturas ilegales de los archivos en los que se almacena. También, en su viaje por el canal de comunicaciones, está expuesto a copias e intervenciones de todo tipo, por parte de entes que pueden ser usuarios o procesos y que se conocen como *intrusos*, *atacantes*, *enemigos*, *adversarios*, etc. Un estudio más detallado sobre este problema, la clasificación y tipos de ataques a estos esquemas se ofrece en la sección 2.3 del Capítulo 2 de este trabajo.

[1] Sección A.4 del Apéndice A

1.3.1 Esquemas de Password Convencional

Los esquemas de password convencional involucran passwords invariantes con respecto al tiempo, los cuales proporcionan lo que se conoce como *autenticación débil*. La idea básica es la siguiente. Un password, asociado con cada usuario o entidad, típicamente es una cadena de 6 a 10 o más caracteres que el usuario es capaz de memorizar. Ese password sirve como secreto compartido entre el usuario y el sistema y el proceso de autenticación se basa en ese secreto.

Para tener acceso a un recurso del sistema (ejemplo: procesador, impresión, o alguna aplicación), el usuario teclea su *"login"* y el password asociado, y explícita o implícitamente especifica un recurso. En este caso, el *"login"* es un reclamo de identidad y el password es la evidencia que soporta tal reclamo. El sistema verifica que el password coincida con los datos correspondientes que mantiene para ese identificador y que la identidad declarada haya sido autorizada para accesar el recurso. La demostración del conocimiento de ese password (revelando el password mismo o alguna función de él) es aceptada por el sistema como corroboración de la identidad del usuario [50] (pp. 388).

Varios de estos esquemas se distinguen entre sí por la forma en que se almacena la información que permite la verificación del password, y por el método de verificación. Algunas de las técnicas de almacenamiento más comunes en estos esquemas, son las siguientes [50] (pp. 389):

1. *Archivos de passwords en claro*

   El esquema de password convencional más obvio es el que almacena los passwords del usuario en claro en un archivo, protegiéndolo contra lectura y escritura por medio de los privilegios de control de acceso del sistema operativo (un ejemplo de este caso es el sistema operativo VM de la arquitectura 370 de IBM). Al momento de autenticar, el sistema compara el password tecleado por el usuario con la entrada correspondiente en el archivo de passwords. Esta técnica se clasifica como no criptográfica. Una debilidad de este método es que no proporciona ninguna protección contra usuarios privilegiados o superusuarios. El respaldo del archivo de passwords también constituye un problema de seguridad debido a que los passwords están en claro.

2. *Archivos de passwords cifrados*

En lugar de guardar los passwords en claro en un archivo, lo que se guarda es el resultado de aplicar una función hash a cada password y se pone en lugar del password mismo. (Una función hash es una transformación matemática que toma como entrada un texto en claro o mensaje en claro de tamaño variable y proporciona una salida de tamaño fijo y pequeño conocida como valor o resultado hash. Para mayor detalle sobre funciones hash, ver la Sección A.2 del Apéndice A). Para verificar el password del usuario, el sistema calcula la función hash del password tecleado y lo compara con el resultado hash de la entrada correspondiente al usuario en el archivo de passwords.

Las funciones hash, junto con las técnicas de llave secreta o simétricas y de llave pública o asimétricas, constituyen una de las técnicas de la criptografía. Los fundamentos de estas técnicas se resumen en la Sección A.2 del Apéndice A des este trabajo.

El usar funciones hash en lugar técnicas de llave secreta es preferible debido a las restricciones que el gobierno de los Estados Unidos de Norteamérica impone a la exportación de este tipo de tecnología y a la necesidad de uso de llaves que las técnicas de llave secreta presentan. Por razones históricas a los valores resultantes de utilizar ambas técnicas sobre passwords se les conoce como passwords cifrados.

1.3.2 Mejoras a Esquemas de Password Convencional

Con el fin de mejorar los esquemas de passwords convencionales, se han propuesto y se utilizan varios mecanismos que tienden a fortalecer la elección del password o a hacer más difícil la acción de un atacante sobre ellos. Los principales mecanismos que se utilizan son los siguientes [50] (pp. 389 - 391):

1. *Restricciones a elección de passwords*

   Debido a que los ataques tipo diccionario (Subsección 2.3.3 del Capítulo 2) son dirigidos contra passwords débiles y predecibles, algunos sistemas imponen restricciones a la manera en que estos se escogen con el fin de desalentar o prevenir la utilización de tales passwords débiles. Las restricciones típicas incluyen: establecer un límite inferior a la longitud del password (ejemplo, 8 o 12 caracteres); el requerimiento de que cada password contenga al menos un caracter de cada uno

de un conjunto de cotegorías (ejemplo, mayúsculas, minúsculas, no alfanuméricas, etc.); verificar que los passwords no se encuentren en diccionarios en línea o disponibles, y que no contengan información relacionada a los *"logins"* del usuario o subcadenas de ellos.

No obstante, conociendo las restricciones operantes, un atacante puede usar un diccionario modificado que tome en cuenta esas restricciones. El objetivo de las restricciones es incrementar la entropía (más que la longitud) de los passwords para dejarlos fuera del alcance de ataques por diccionario y búsqueda exhaustiva (Subsección 2.3.3 del Capítulo 2). Formalmente, entropía es una medida $H()$ de la cantidad de información del mensaje M, denotada como $H(M)$; la cantidad de información en un mensaje se define como el mínimo número de bits necesarios para codificar todos los posibles significados de ese mensaje. Por ejemplo, para representar el día de la semana en un campo de una base de datos se necesitan no más de 3 bits de información; para representar el sexo de una persona se necesita sólo 1 bit. La entropía en este caso es 1 para el sexo, y una cantidad ligeramente menor que 3 para el día de la semana. En general, la entropía de un mensaje medido en bits es $log_2 n$ donde n es el número de posibles significados, asumiendo que cada significado es igualmente probable [38] (pp. 233 - 234).

En este contexto, entropía se refiere a la incertidumbre acerca del conocimiento sobre un password (Sección A.9 del Apéndice A); si todos los passwords son igualmente probables, entonces la entropía es máxima e igual a $log_2 n$ del número de n posibles passwords [50] (pp. 389).

2. *Establecer una edad para el password*

   Otro procedimiento que se utiliza para mejorar la seguridad de los passwords es darles una edad, definiendo un periodo de tiempo que limita la vida de cada password particular (ejemplo, 30 o 90 días). Esto obliga a que los passwords se cambien periodicamente.

3. *Mapeo lento de passwords*

   Una forma de desalentar un ataque que involucra probar un gran número de passwords (Subsección 2.3.3 del Capítulo 2) es hacer lento el proceso de verificación. Un modo de lograrlo es que la función de verificación (ejemplo, una función hash) del password sea computacionalmente intensiva. Por ejemplo, iterando una función más simple

t veces, donde la salida de la iteración i se use como entrada para la iteración $i + 1$. El número total de iteraciones debe restringirse de tal manera que no imponga retardo notable a los usuarios legítimos. También, la función iterada debe ser tal que el mapeo iterado no dé como resultado un rango final cuya entropía resulte significativamente menor.

4. *Alteración del sabor del password ("salting")*

   Para hacer que los ataques por diccionario sean menos efectivos, cada password puede aumentarse con una cadena aleatoria de t bits llamada *sal*, alterando con esto el *sabor* del password (Subsección 3.1.1 del Capítulo 3) antes de aplicar la función hash. Ambos, el valor hash del password y la sal se guardan en el archivo de passwords. Cuando el usuario teclea su password, el sistema verifica la sal, y aplica la función al password alterado o aumentado por la sal. La dificultad para una búsqueda exhaustiva de cualquier password en particular no cambia con la sal (debido a que la sal aparece en claro en el archivo de passwords); no obstante, la sal incrementa la complejidad de un ataque por diccionario contra un conjunto grande de passwords simultáneamente, porque necesita que el diccionario contenga 2^t variaciones de cada password que se prueba, lo cual requiere de gran cantidad de memoria para guardar un diccionario cifrado, y correspondientemente, más tiempo para su preparación. Con la sal, dos usuarios que escojan el mismo password tendrán, muy probablemente, entradas distintas en el archivo de passwords. En algunos sistemas, puede ser apropiado usar el *"login"* del usuario como sal [50] (pp. 390).

5. *Uso de frases como passwords*

   Para lograr mayor entropía sin rebasar la capacidad de memoria de los humanos, los passwords pueden extenderse a frases que se usan como tales. En este caso, el usuario teclea una frase o sentencia en lugar de una palabra corta. Se toma el valor hash de la frase, reduciéndola de este modo a un valor de tamaño fijo, que juega el mismo papel que el password. Es importante que la frase no sea simplemente truncada por el sistema, como lo son los passwords en algunos sistemas como UNIX (Subsección 3.1.1 del Capítulo 3). La idea es que los usuarios puedan recordar frases más fácilmente que secuencias de caracteres aleatorios. Si los passwords se asemejan a texto en inglés, entonces, ya que cada caracter contiene solamente cerca de 1.5 bits de entropía [50] (pp. 246), una frase proporciona mayor seguridad, por medio de

una entropía incrementada, que un password corto.

1.3.3 Passwords de una sola vez (hacia Autenticación Fuerte)

Un problema de los esquemas de password convencional es el monitoreo en la línea de comunicaciones por parte de atacantes pasivos que capturan *los passwords, ya sea que viajen cifrados o no, y posteriormente intentan* suplantar a una de las partes legítimas construyendo mensajes réplicas con los passwords capturados.

Una mejora natural a los esquemas de password convencional consiste en esquemas de passwords que se utilizan una sola vez. Las variaciones [50] (pp. 395 - 396) de este esquema incluyen:

1. *Listas compartidas de passwords de única vez*

   En esta variante el usuario y el sistema usan una secuencia o conjunto de t passwords (cada uno válido para una única autenticación), que se distribuyen como una lista previamente compartida. Un inconveniente es el mantenimiento de esa lista compartida. Si la lista no se usa secuencialmente, el sistema puede verificar el password tecleado contra todos los restantes passwords sin usar.

   Una variación involucra el uso de una tabla de parejas de elementos relacionados que el usuario y el sistema comparten. Típicamente un elemento de la tabla, llamado *quienvive*, es un número aleatorio que cambia con respecto al tiempo, por lo que se genera y se utiliza una única vez. El otro elemento, llamado *respuesta*, depende tanto del quienvive como del password, y puede ser el resultado de una transformación tipo hash o de algún otro tipo. Cada pareja en la tabla debe ser válida al menos una vez. Esta es una técnica no criptográfica. Existen también técnicas criptográficas que utilizan intercambios quienvive-respuesta (Sección 1.4 del Capítulo 1).

2. *Passwords de única vez secuencialmente actualizados*

   En esta variante, inicialmente se comparte un único password. Durante la autenticación usando el password i, el usuario crea y trasmite al sistema un nuevo password $(i + 1)$, cifrado con una llave derivada

del password i. Una debilidad de este método es que se hace difícil si ocurre una falla en la comunicación.

3. *Secuencias de passwords de única vez basadas en una función hash*

   El esquema de password de una sola vez de Lamport [50] (pp. 396) es un ejemplo de esta variación. La parte A inicia eligiendo un password w y un número constante razonablemente grande t, que define el número de identificaciones permitidas (ejemplo, $t = 100$ o 1000). Usa una función hash h para calcular: $h(w)$, $h(h(w))$,...,$h^t(w)$. Envía t y $h^t(w)$ a la parte B. Cuando A desea autenticarse ante B, le envía su identidad. B le regresa t. A calcula $h^{t-1}(w)$ y lo envía a B. Este calcula $h(h^{t-1}(w))$ y lo compara con $h^t(w)$ que ha mantenido guardado. Si hay coincidencia, B considera válida la respuesta y reemplaza $h^t(w)$ con $h^{t-1}(w)$, y t con t - 1.

   En este esquema, el password para la i-ésima sesión de identificación, donde $1 \leq i \leq t$, está definido por $w_i = h^{t-i}(w)$. Este método es más eficiente que la actualización secuencial, donde en cada proceso de autenticación se debe crear y trasmitir un nuevo password cifrado con una llave derivada del password usado en el proceso de autenticación previo.

   El esquema de Lamport puede verse como un protocolo quienvive-respuesta, donde el quienvive está implícitamente definido por la posición actual dentro de la secuencia de passwords.

1.4 Autenticación Fuerte

La idea de los protocolos llamados fuertes es que una entidad (la que se identifica) pruebe su identidad a otra entidad (la que autentica) demostrando el conocimiento del password asociado con esa entidad, sin revelar el password mismo a la parte que autentica durante el protocolo. (En algunos mecanismos, el password es conocido por la parte que autentica y es usado para verificar la respuesta de la parte que se identifica; en otros, el password no necesita realmente ser conocido por la parte que autentica) [50] (pp. 397).

Lo anterior se hace proporcionando una respuesta a un quienvive que cambia con respecto al tiempo, donde la respuesta depende tanto del password como del quienvive. El quienvive es típicamente un número escogido por

una entidad (de manera aleatoria y secreta) al inicio del protocolo. Si la línea de comunicaciones es monitoreada, la respuesta de una ejecución del protocolo no debe proporcionar información útil a un atacante para una autenticación posterior, ya que los quienvives subsecuentes serán distintos. Existen protocolos de este tipo basados en técnicas de llave simétrica, de llave pública, y en conceptos de conocimiento cero [50] (pp. 400 - 417). Las técnicas criptográficas de llave simétrica, de llave pública, y funciones hash se revisan en la Sección A.2 del Apéndice A de este trabajo. El concepto de conocimiento cero se trata en la Sección A.8 del mismo apéndice.

1.5 Protocolos Criptográficos de Autenticación

La búsqueda de esquemas de seguridad que incluyan, entre otros servicios, autenticación segura, ha llevado a la necesidad de definir de manera más precisa la parte que, dentro de ese esquema general, corresponde a cada servicio de seguridad, y poderlo estudiar de manera independiente de los otros servicios. Surge de esta manera el concepto de *protocolo*, que informalmente hablando, es una serie de pasos que involucran a dos o más partes y se diseña para realizar una tarea específica [38] (pp. 21).

De manera más formal, un protocolo consiste en una especificación para el formato y regulación de ocurrencias en tiempos relativos a la información intercambiada entre las partes que se comunican [2].

La búsqueda de esquemas específicos de autenticación se reduce, de este modo, a la búsqueda de protocolos de autenticación para una categoría específica de autenticación, utilizando tal o cual herramienta. Por ejemplo, como en este trabajo, el estudio de protocolos de autenticación basados en passwords que utilizan técnicas criptográficas.

Así, un *protocolo criptográfico* es un protocolo de comunicación que utiliza *criptografía*[3] para garantizar la integridad, confidencialidad, origen, destino, orden, puntualidad en el tiempo de llegada y, de esta forma, el significado de los mensajes intercambiados entre los participantes en el protocolo, sobre

[2] *Dictionary of Computers, Data Processing, and Telecommunications* Jerry M. Rosenberg, John Wiley and Sons, 1984, p.411.

[3] Sección A.1 del Apéndice A

un medio de comunicación inseguro que puede estar intervenido por intrusos hostiles, los cuales pueden intentar subvertir los objetivos del protocolo [44] (pp. 5).

Todos los protocolos de autenticación basados en passwords, tratan de resolver el mismo problema: una parte debe, de alguna manera, probar a la otra parte que conoce el password P, usualmente establecido con anterioridad [11] (pp. 1). Tales protocolos varían desde lo trivial a lo bastante complejo, y todos ellos ofrecen alguna forma de protección contra varios tipos de ataques montados por intrusos maliciosos, o excesivamente curiosos.

Este tipo de protocolos tienen un amplio rango de aplicaciones prácticas debido a que no requieren nada más que un password memorizado. Estas aplicaciones van desde el acceso remoto a computadoras, hasta aplicaciones en *Internet* [12].

Existen muchos protocolos de este tipo implementados en esquemas de autenticación actualmente en uso. Ejemplos de esquemas de autenticación ampliamente utilizados que implementan protocolos de esta clase son, entre otros:

1. El sistema operativo $UNIX$ [2, 13, 15] utiliza un esquema de password almacenado, cifrado con una función hash unidireccional [4]. A pesar de ser ampliamente usado y conocido, este sistema operativo adolece de debilidades en su esquema de autenticación. $UNIX$ y su problemática de seguridad se estudian con amplitud en la Sección 3.1 del Capítulo 3 de este trabajo.

2. El sistema de autenticación de Kerberos [4] desarrollado en el M.I.T. (*"Massachusetts Institute of Technology"*) está basado en la familia de protocolos de Needham y Schroeder [26, 27, 32]. No obstante su popularidad, Kerberos ha sido objeto de críticas, tanto por los protocolos en que se basa como por su implementación [33].

3. El sistema de seguridad en red conocido como KryptoKnight [35] implementa protocolos de tres partes para autenticación y distribución de llaves [34] basados en passwords.

[4] Sección A.2 del Apéndice A

Muchas de los esquemas anteriores, aunque están actualmente en uso, han mostrado ser inseguros. En la Sección 3.5 del Capítulo 3 de este trabajo se hace una revisión de estos esquemas.

A partir 1992, la investigación de protocolos criptográficos basados en passwords se vió fortalecida debido a los trabajos de Bellovin y Merrit [7, 17] y de Gong, Lomas, Saltzer, y Needham [1, 6], grupos que de manera separada presentaron protocolos que superaban con mucho a los hasta entonces existentes. Surge de este modo la vertiente de investigación de lo que se conoce como protocolos fuertes. En el capítulo 4 de este trabajo se hace un análisis de estos protocolos, presentando los más representativos.

En la actualidad existen dos direcciones de investigación claramente definidas en el área de protocolos de autenticación e intercambio de llaves: una que busca lograr sus objetivos basada en el hecho de que el password o la llave o llaves no viajan por el canal de comunicaciones, por lo que no hay nada que proteger usando alguna técnica criptográfica; y otra que busca los mismos objetivos a partir del hecho de que el password o la llave o llaves viajan por el canal y debe ser protegido en su viaje, utilizando alguna técnica criptográfica. En el capítulo 3 de este trabajo se abunda más sobre estas vertientes de investigación.

1.6 Estructura del Trabajo

El resto de este trabajo está dividido como sigue: en el capítulo 2, se presenta la especificación del problema de la forma más general posible, las suposiciones, los posibles ataques, el ambiente, las métricas, y la notación.

En el capítulo 3 se revisa, por su importancia en el contexto actual, la problemática de seguridad de *UNIX* hasta la fecha; se presentan las alternativas basadas en passwords que para autenticación e intercambio de llaves se han utilizado hasta ahora; se presenta una revisión de los protocolos de intercambio de llaves y su relación con el proceso de autenticación; finalmente, se presentan las vertientes de investigación que se siguen en los protocolos llamados fuertes y el estado del arte actualmente.

En el capítulo 4 se describen y analizan los protocolos más representativos

de cada una de las vertientes criptográficas, sus características, fortalezas y debilidades.

En el capítulo 5, se hace un análisis comparativo de los protocolos fuertes presentados en este trabajo. En este capítulo se empieza por hacer una revisión de los distintos protocolos que se han propuesto hasta la fecha para resolver el problema de autenticación basada en password, resumiendo las principales referencias sobre el tema. Después, se definen las características básicas que deben cumplir los protocolos fuertes y otras que idealmente deberían cumplir; estas características son la base del análisis que se hace. El análisis comparativo se realiza de acuerdo a las características definidas, suposiciones, ataques que soportan, ventajas, y desventajas de cada uno de los protocolos analizados. El resultado de este análisis se resume en una tabla comparativa que se muestra en la subsección 5.2.7 de este capítulo. También, en la sección 5.3, se presenta una estructura arborescente del desarrollo de estos protocolos en el tiempo, sus influencias y derivaciones. En las últimas secciones de este capítulo, se presenta información sobre implementaciones existentes, estudios de eficiencia, y aplicaciones actuales y futuras de protocolos fuertes.

En el capítulo 6 se presentan los resultados más importantes de este trabajo, las conclusiones que se derivan de él y el trabajo futuro que puede delinearse a partir del presente estudio en el área de protocolos de autenticación fuerte basados en passwords.

En el Apéndice A se presenta una breve introducción a conceptos básicos relacionados con este trabajo. La sección A.1 trata los conceptos de criptografía y criptosistema, entre otros. En la sección A.2 se resumen las técnicas criptográficas de llave secreta, llave pública, y funciones hash. En la sección A.3 se definen el concepto de criptoanálisis y se describen los principales ataques por criptoanálisis. En la sección A.4 se enfatizan las suposiciones criptográficas usadas en autenticación.

La sección A.5 trata brevemente el concepto de firma digital; la sección A.6 trata sobre el sistema de seguridad conocido como S/KEY; la sección A.7 hace lo propio con el sistema de propósito general usado actualmente en algunos navegadores, conocido como SSL(*"Secure Socket Layer"*). En la sección A.8 se resume el concepto de Pruebas por Conocimiento Cero, en A.9 se formaliza el concepto de entropía, en A.10 se tratan brevemente los conceptos de grupos, subgrupos y cociente de Euler; en A.11 se formaliza

el algoritmo de acuerdo de llave Diffie-Hellman mostrando el protocolo base y definiendo los problemas conocidos como problema Diffie-Hellman y el problema de logaritmos discretos.

En el Apéndice B se presenta una relación referencial sobre las principales fuentes de información disponible acerca de protocolos fuertes de autenticación e intercambio de llaves, y temas relacionados. En la sección B.1 se listan direcciones electrónicas donde se puede hallar información reciente en línea sobre protocolos fuertes. La sección B.2 lista direcciones de interés sobre conferencias, cursos, seminarios, libros, revistas, notas y periódicos sobre temas relacionados. En la sección B.3 se listan las principales revistas que publican artículos sobre protocolos fuertes y áreas relacionadas. En la sección B.4 se mencionan los principales simposios, conferencias y *"workshops"* sobre el tema y áreas similares. Finalmente, la sección B.5 presenta los principales *"proceedings"* de conferencias, simposios, y *"workshops"* sobre el tema y áreas afines.

Capítulo 2

Especificación del Problema

En este capítulo se describe el planteamiento del problema motivo de este trabajo, se presenta el modelo que sirve para definir de manera precisa el escenario, los participantes legítimos en el protocolo, el atacante, entre otros componentes. También se precisan las suposiciones para el estudio del problema, los parámetros de ambiente, las métricas, y la notación que se utiliza a lo largo del trabajo. Igualmente, en este capítulo se describen los principales ataques a que están expuestos los protocolos de autenticación basados en passwords, las estrategias de ataque, su clasificación, y los escenarios donde se llevan a cabo.

2.1 Planteamiento del Problema

En un sistema de seguridad que basa su proceso de autenticación en passwords seleccionados por los usuarios, estos escogen passwords cortos, fáciles de memorizar y, normalmente relacionados con nombres de personas, animales, cosas, ciudades, etc. que tienen que ver con el universo de cada usuario y que existen en un diccionario del idioma que se trate o bien se puede fácilmente construir una lista de posibles passwords a partir de ese diccionario, combinando palabras en el diccionario con números, fechas, caracteres especiales, etc. También pueden agregarse a esa posible lista nombres propios, de canciones, de películas, etc. Pueden crearse listas tan

sofisticadas como se desee a partir de saber la forma en que los usuarios tradicionalmente eligen sus passwords. Los passwords así elegidos son criptográficamente débiles y fáciles de atacar.

Se dice que esos passwords son criptográficamente débiles porque el espacio de todos los posibles passwords es pequeño, y un atacante puede intentar un ataque por *fuerza bruta* barriendo ese espacio en un tiempo computacionalmente viable, hasta dar con el password correcto. Aún peor, se pueden atacar a un costo menor porque un atacante puede reducir el espacio de todos los posibles passwords construyendo diccionarios de passwords probables y probando todos ellos hasta dar con el correcto. Este último ataque se le conoce como *ataque por diccionario* y se describe con detalle en la Subsección 2.3.3 del Capítulo 2 de este trabajo.

Los sistemas tradicionales que basan en passwords su servicio de autenticación, almacenan los passwords en claro (un ejemplo de este caso es el sistema operativo VM de IBM, arquitectura 370), o cifrados de algún modo (ejemplo de este otro caso es el sistema operativo $UNIX$), y utilizan esa información para su proceso de autenticación.

En esos sistemas tradicionales, los passwords almacenados pueden ser sustraídos de alguna manera por algún atacante, el cual puede ser un usuario legítimo o ilegítimo del sistema, y comprometer la seguridad del propio sistema y de los usuarios, ya sea enterándose directamente de los passwords en caso de que se almacenen en claro, o montando un ataque diccionario en caso de que se guarden cifrados. Esta problemática involucra el resguardo y seguridad de los archivos que contienen los passwords en claro o cifrados. Pero también, si se guardan cifrados, tiene que ver con el hecho de que los passwords sean criptográficamente débiles.

También la seguridad del sistema puede comprometerse durante el proceso mismo de autenticación, durante el cual la información sensible tal como la identidad y el password, que se intercambian entre los participantes en un protocolo tradicional, viaja en claro por el canal de comunicaciones inseguro, y está expuesta a intervenciones de todo tipo.

Si el password viaja en claro, el atacante sólo tiene que monitorear el canal y capturar el password en su viaje por el canal. Si viaja cifrado y la llave de cifrado es el password o una llave derivada algoritmicamente de él, el atacante captura el password cifrado y luego intenta adivinar el password

en claro probando posibles passwords mediante un ataque diccionario. Para lograr esto, aplica a los candidatos el mismo algoritmo de cifrado que utiliza el protocolo, que es público y conocido, y compara el resultado de ese proceso con los passwords cifrados que tiene en su poder. Si hay coincidencia, el atacante ha adivinado el password correcto.

El problema consiste en **diseñar protocolos de autenticación que permitan que un participante legítimo del protocolo pueda probar a la otra parte su conocimiento sobre un password sin revelar ese password a nadie, ni siquiera a la parte con la que se está comunicando.** Este problema pudiera parecer imposible de resolver porque aún utilizando criptografía para cifrar el password en su viaje por el canal, tratándose de un password criptográficamente débil, está expuesto a intervenciones y ataques.

Existen algunas alternativas de solución actualmente en uso pero no son satisfactorias, tal como obligar a los usuarios a usar passwords criptográficamente fuertes (Subsección 1.3.2 del Capítulo 1). Otras alternativas en uso actual tienen debilidades de seguridad, como las que usan criptografía pero permiten ataques diccionario y su seguridad depende de no comprometer el archivo donde se guardan los passwords cifrados [33, 52].

Existen soluciones al problema que no se usan o se usan poco en implementaciones actuales de servicios de seguridad porque son ineficientes en términos computacionales, como algunas de las llamadas *soluciones fuertes* [11]. Otras soluciones están en proceso de investigación y perfeccionamiento.

El presente trabajo, como lo hemos mencionado en el capítulo anterior, se enfoca a estudiar y analizar soluciones criptográficas fuertes.

2.2 El Modelo

Los protocolos criptográficos de autenticación basados en passwords, normalmente trabajan sobre un esquema *cliente/servidor* en una red de computadoras o en un sistema distribuido, donde el *cliente*, también llamado *demandante*, puede ser un usuario, una computadora, un proceso o algún otro dispositivo; y el *servidor*, también conocido como *verificador o autenticador*, casi siempre es una computadora, aunque puede ser un proceso u

otro dispositivo. Comúnmente, al servidor también se le llama *"host"*.

De manera general, como participantes en el protocolo, tanto al cliente como al servidor se les conoce como *principales*. Los principales intercambian información a través de un canal de comunicaciones público inseguro en la red o sistema distribuido.

En términos de un modelo preciso para el estudio de protocolos, autenticación es el proceso mediante el cual un participante en el protocolo se asegura, a través de la adquisición de evidencia corroborativa, de la identidad de una segunda parte involucrada en el protocolo; también se asegura de que esa segunda parte haya participado realmente en el protocolo. Es decir, se asegura que esa segunda parte esté activa, o lo haya estado inmediatamente antes del momento en que la evidencia corroborativa fue adquirida [50] (pp. 386).

El modelo para estudiar los protocolos criptográficos de autenticación basados en passwords consiste de los participantes A, que funciona como cliente, y B que lo hace como servidor. A tiene conocimiento sobre un password P que normalmente memoriza, o aprende, o guarda en algún lugar que sólo él sabe y que usa al momento de solicitar un servicio a B. P le sirve a A para probar su identidad ante B, a través de un *protocolo de autenticación*, que ambas partes ejecutan. B, puede conocer explícitamente P o no. Lo conoce si lo mantiene guardado en algún lugar de su memoria y lo usa para verificar la identidad de A.

También puede ser que lo que guarde B no sea el P de A tal cual, sino alguna información derivada de él. Por ejemplo, puede guardar el resultado de aplicar una transformación matemática computacionalmente infactible de invertir a P, tal como una función hash [1] por ejemplo. Esto significa que B no conoce ni puede conocer P en claro. En todo caso, B es capaz de derivar, ejecutando algún proceso basado en la información que recibe de A, el resultado que guarda. Al comparar el resultado de tal proceso con el que guarda, B se convence de la identidad de A.

Cualquiera que sea el esquema de conocimiento sobre P, A y B pueden ejecutar instancias del protocolo acordado, donde cada instancia es independiente de las otras. A y B pueden hacer varias ejecuciones del protocolo

[1] Sección A.2 del Apéndice A

si desean interactuar sobre varias conexiones autenticadas paralelas.

Desde el punto de vista de B (la parte que autentica), el resultado del protocolo es la *aceptación* de la identidad de A (la parte autenticada) como auténtica (finalización del protocolo con aceptación), o la terminación sin aceptación o *rechazo*. Más específicamente, los objetivos de un protocolo de autenticación son los siguientes [50] (pp. 386 - 387):

1. En el caso de partes honestas A y B, A es capaz de autenticarse él mismo ante B. Es decir, B terminará la ejecución del protocolo habiendo aceptado la identidad de A
2. B no puede reutilizar un intercambio de mensajes de autenticación con A para suplantar exitosamente a A ante una tercera parte I
3. La probabilidad es despreciable de que cualquier parte I distinta de A, ejecutando el protocolo y jugando el papel de A, pueda provocar que B termine la ejecución aceptando la identidad de A
4. El punto anterior es cierto aún si: a) un gran número (polinomial) de autenticaciones previas entre A y B han sido observadas; b) el atacante I ha participado en ejecuciones previas del protocolo con A, con B, o con ambos; c) múltiples instancias del protocolo, posiblemente iniciadas por I, pueden ejecutarse simultáneamente

Un protocolo de autenticación es un proceso en *tiempo real* en el sentido que proporciona una certeza de que la parte que está siendo autenticada está en operación en el momento de la ejecución del protocolo. Es decir, esa parte está tomando parte o llevando a cabo alguna acción desde el inicio de la ejecución del protocolo. El protocolo proporciona certezas sólo en el momento particular en que finaliza exitosamente su ejecución [50] (pp. 387).

Un atacante, I, de un protocolo de autenticación puede describirse intuitivamente como una tercera parte que no tiene acceso al password. No obstante, tiene acceso a todos los mensajes intercambiados en cada una de las ejecuciones legítimas del protocolo aceptadas por A y B en el pasado; y es capaz, en cualquier momento, de interferir con ejecuciones en marcha del protocolo, o iniciar nuevas ejecuciones, que involucren a A, a B, o a ambas.

El objetivo del atacante es provocar que A o B registren erróneamente una

de las ejecuciones ilegítimas del protocolo como aceptada, aún cuando no coincida con ninguna ejecución del protocolo aceptada por la otra parte.

De lo anterior se desprende que el atacante I, puede adaptivamente enviar cualquier mensaje a cualquiera de las partes; puede iniciar nuevas ejecuciones del protocolo con A o con B haciéndose pasar por una de las partes legítimas; y puede interceptar mensajes enviados por una u otra parte.

No se imponen restricciones de tiempo de respuesta sobre las acciones anteriores. Los retardos son arbitrarios, pero se supone que no hay pérdida de mensajes. De garantizar esto último se encargan las capas inferiores al nivel donde se ejecuta el protocolo de autenticación.

2.2.1 Suposiciones

Estos protocolos asumen que la red entre A y B es vulnerable a intromisiones y falsificaciones deliberadas por parte de un atacante (Sección 2.3 del Capítulo 2). También se asume que cualquier principal puede poner o introducir un mensaje sobre cualquier línea en cualquier momento; puede ver todos los mensajes intercambiados; puede borrar, alterar o redireccionar cualquier mensaje que pase sobre cualquier línea; puede iniciar comunicaciones con cualquier otra parte; y puede replicar mensajes registrados, pertenecientes a comunicaciones pasadas entre las partes legítimas.

Aunque se analizarán algunos casos importantes de autenticación indirecta (Sección 1.2 del Capítulo 1), el presente trabajo, se enfocará a los protocolos criptográficos para autenticación directa (Sección 1.2 del Capítulo 1) basados en passwords. Es decir, se asume que no hay una tercera parte en la que los participantes en el protocolo puedan confiar, tal como un servidor de llaves o una autoridad certificadora; sólo existen los dos participantes originales en el protocolo.

Precisando más la participación de las dos únicas partes en el protocolo: se asume que A es siempre el originador de la ejecución del protocolo y B responde siempre a una petición.

Al iniciar la ejecución del protocolo, se asume que las partes comparten el conocimiento sobre P de alguna de las formas descritas en el modelo. Al ter-

minar la ejecución del protocolo, si esta ejecución fue exitosa, ambas partes se habrán asegurado que cada una es quien dice ser, que nadie más puede fácilmente hacerse pasar por una de las partes, y al final comparten una llave temporal de sesión, útil para proteger únicamente esa sesión. Se asume que la llave temporal se usa en algoritmos criptográficos convencionales.

Así, autenticación aquí significa identificación más intercambio de llave de sesión. Esto es así por la conveniencia de juntar en un sólo proceso lo que separadamente serían dos protocolos y porque al ejecutarse juntos se hace más eficiente y seguro. La idea básica de esta integración es que separar los pasos de autenticación e intercambio de llaves crea oportunidades para ataques de tipo hombre enmedio [19] (pp. 3). Este ataque se describe con detalle en la Subsección 2.3.4 del Capítulo 2 de este trabajo.

En un protocolo de autenticación, que también proporciona intercambio de llaves, los participantes además de probar su identidad, desean distribuir entre sí una llave de sesión *fresca* y *secreta*. Se dice que una llave es *fresca*, desde el punto de vista de una de las partes, si se puede garantizar que es nueva, es decir, recientemente enviada y generada; esto evita la posibilidad que una llave vieja o usada con anterioridad pueda ser de nuevo utilizada por un atacante o por una parte legítima.

A primera vista, pudiera parecer innecesario que las partes que ya comparten un secreto, el password, requieran otro. Una razón de su utilidad es la necesidad de evitar ataques tipo *réplica* en sesiones cruzadas. Este ataque consiste en utilizar mensajes registrados en una cierta sesión y hacerlos pasar por auténticos en otra [24]. Esto podría pasar si no se contemplara la utilización de una llave temporal de sesión como un segundo secreto compartido, distinto al password.

Respecto a lo anterior, un protocolo de autenticación usualmente cae en uno de dos niveles respecto a sus objetivos [23]. Un protocolo en el primer nivel puede verse como sólo autenticación e intercambio de llaves. Esto significa que al finalizar la ejecución del protocolo, cada participante habrá recibido una llave fresca y el cliente cree que esa llave es adecuada para la comunicación con el servidor; es decir, la llave fue enviada sin haber recibido confirmación de recepción.

Un protocolo en el segundo nivel puede verse como autenticación e intercambio de llaves con *saludo*. Esto significa que, al finalizar la ejecución del

protocolo, cada participante también cree que la otra parte ha recibido la llave correctamente y cree en lo adecuado de esa llave. Este requerimiento extra de confirmación se satisface generalmente por medio de una serie de mensajes que se intercambian al final del protocolo. Este intercambio final es el que se conoce como *saludo* o *"handshake"* [10] (pp. 3).

2.2.2 Parámetros de Ambiente

Cuando se utilizan herramientas criptográficas en los protocolos de autenticación e intercambio de llaves, hay dos parámetros de ambiente importantes a considerar [10] (pp. 4). El primero de estos parámetros tiene que ver con la selección de la llave de sesión entre las partes involucradas en el protocolo. El segundo parámetro es un elemento que depende del tiempo y que se conoce como quienvive, cuestionamiento o reto. En algunos protocolos este parámetro se envía como requerimiento y se espera una respuesta que depende de ese parámetro y del password. Estos protocolos se conocen como del tipo quienvive-respuesta. (Sección 1.4 del Capítulo 1).

1. El primer parámetro es concerniente a quién de las partes involucradas en el protocolo escoge la llave temporal de sesión. Generalmente, se consideran tres posibilidades: la primera es que la parte que actúa como servidor la escoja; la segunda es que la escoja cualquiera de las dos partes; y la última es que ambas partes participen en la selección de la llave. Dejar al cliente la selección de la llave, con frecuencia reduce el número de mensajes y rondas del protocolo [10] (pp. 4).

2. Los parámetros dependientes del tiempo o quienvives pueden usarse en los protocolos de autenticación para contrarrestar ataques tipo réplica y combinación (Subsección 2.3.4 del Capítulo 2), para proporcionar unicidad o puntualidad, y para prevenir ciertos ataques por texto escogido (Subsección 2.3.4 de este capítulo) [50] (pp. 397).

   El concepto de unicidad tiene que ver con la necesidad de no intercambiar exactamente el mismo mensaje cifrado varias veces. Si hubiera repetición, un atacante podría explotar esa redundancia en su favor y montar un ataque, por ejemplo, tipo réplica. El concepto de puntualidad tiene que ver con el hecho de que los mensajes intercambiados entre las partes sean enviados y recibidos dentro de un cierto rango

de tiempo definido por esas partes, y controlado por los relojes locales del enviador y del receptor. Este concepto se tratará con detalle más adelante en esta misma sección.

Los parámetros dependientes del tiempo o quienvives sirven para distinguir distintas ejecuciones de un protocolo, garantizando la *frescura* de un mensaje. Es decir, garantizando que se trata de un mensaje con un elemento de unicidad (el quienvive) recientemente generado que no se volverá a utilizar en esa ejecución del protocolo. Algunas veces estos parámetros se conocen como *"nonces"*, *números únicos*, o *valores no repetidos*. En este trabajo los llamaremos genéricamente *núnicos*, aún cuando en el caso de lecturas directas del reloj de la computadora se les seguirá mencionando como instantáneas (*"timestamps"*).

Estos parámetros son de diversos tipos y tienen distintos nombres de acuerdo, generalmente, a la función o funciones que desempeñan. Pueden servir para garantizar unicidad de los mensajes, en cuyo caso se conocen como *núnicos* (*"nonces"*); otras veces se usan para garantizar unicidad y puntualidad, en cuyo caso se conocen como *instantáneas*; finalmente también se utilizan para realizar la etapa del saludo y se conocen como *quienvives*.

A continuación se discuten tres principales clases de estos parámetros: números aleatorios, números en secuencia o serie, e instantáneas.

Con frecuencia, para proteger la seguridad del protocolo, debe garantizarse la integridad de tales parámetros (por ejemplo, ligándolos criptográficamente con otros datos en una secuencia quienvive-respuesta. Esto significa, por ejemplo, que el parámetro puede ser concatenado a los datos o mensaje para cifrarlos juntos). Esto es particularmente cierto para protocolos en los que el único requerimiento del parámetro es la unicidad, como el proporcionado por un contador secuencial nunca repetido [2] [50] (pp. 398).

(a) *Números Aleatorios*

La idea de número aleatorio en este contexto es generarlo como valor que no se use más de una vez para el mismo propósito. Sirve típicamente para impedir réplicas indetectables y para proporcionar impredecibilidad que evite, por ejemplo, ataques por texto

[2] Estos parámetros predecibles difieren de los números en secuencia o serie en que pueden no estar limitados a cualquier estado almacenado. Sin un ligado criptográfico apropiado, un problema potencial es un ataque *"pre-play"* en el cual el atacante obtiene el parámetro antes de que sea legítimamente enviado.

escogido (Subsección A.3.1 del Apéndice A). El término núnico es muy frecuentemente usado para referirse a un número aleatorio en un protocolo quienvive-respuesta, pero las propiedades de aleatoriedad requerida pueden variar [50] (pp. 397-398).

El concepto de número aleatorio, cuando se usa en el contexto de protocolos de autenticación, incluye números pseudoaleatorios impredecibles para un atacante; esto difiere de aleatoriedad en el sentido estadístico tradicional. En estos protocolos "seleccionar un número aleatorio" significa "generar un número con una distribución computacionalmente indistinguible con respecto a una distribución uniforme, de un espacio muestra especificado" o bien "seleccionarlo de una distribución uniforme" [50] (pp. 398).

Este tipo de núnicos se usan en los protocolos quienvive-respuesta de la manera que sigue. Un participante selecciona e incluye un nuevo núnico en un mensaje de salida. Un mensaje de entrada subsecuentemente recibido (el siguiente mensaje de la misma ejecución del protocolo), cuya construcción requiere el conocimiento de este núnico, y al cual está inseparablemente ligado, se juzga entonces como *fresco* basándose en el razonamiento de que el núnico relaciona los dos mensajes. Esta relación infalsificable se requiere para impedir la posibilidad de concatenar el núnico a un mensaje anterior.

En el contexto de este tipo de protocolos, *frescura* significa reciente en el sentido de haber sido originado subsecuentemente al inicio de la ejecución actual del protocolo [50] (pp. 398 - 399).

Esta clase de núnicos tienen la desventaja de que para generarlos se requiere una semilla inicial con suficiente entropía, y este hecho representa un posible punto de ataque al protocolo. Cuando estos núnicos se usan en protocolos quienvive-respuesta, típicamente el protocolo involucra un mensaje adicional y el enviador del núnico debe mantener información de estado hasta que la respuesta sea verificada.

(b) *Secuencia o Serie de Números*

Este tipo de núnico es un número en secuencia o serie que sirve como número único que identifica un mensaje, y se usa típicamente para detectar réplica de mensajes. Estos números son específicos para una pareja de participantes, y deben ser explícita o implícitamente asociados con el originador y el receptor de un mensaje. Normalmente se requieren números distintos para mensajes de A

a B y de B a A.

Los participantes siguen una política predefinida para numerar mensajes. Un mensaje es aceptado por un participante sólo si el número asociado a él no ha sido usado previamente (o no ha sido usado dentro de un periodo de tiempo especificado), y satisface la política acordada. La política más simple es que una secuencia de números inicie en cero, y se incremente secuencialmente, y cada mensaje sucesivo tenga un número mayor que el mensaje recibido previamente [50] (pp. 399).

El usar secuencias de números tiene la desventaja de que cada parte que se identifica debe registrar y mantener información de estado acerca de cada posible entidad que verifica esa identidad. Esto es para poder determinar previamente los números en la secuencia que ya han sido usados y los que aún son válidos.

(c) *Instantáneas ("Timestamps")*

Las instantáneas también se usan para proporcionar puntualidad y garantizar unicidad, para detectar réplicas de mensajes, para implementar privilegios de acceso limitado, y para detectar retardos forzados.

Las instantáneas son lecturas del tiempo del reloj local de una máquina o dispositivo. Funcionan como sigue. Un participante en el protocolo que origina un mensaje obtiene una instántanea de su reloj local y lo liga criptográficamente al mensaje originado. Esto significa que la instantánea puede ser concatenada al mensaje para cifrarlos juntos. Al recibir ese mensaje, la otra parte obtiene el tiempo actual de su propio reloj y le resta a ese tiempo la instantánea que recibió. El mensaje recibido se considera válido si: 1) la diferencia de tiempo está dentro del rango de aceptación (un intervalo de tiempo fijo seleccionado y acordado entre las partes para incluir el máximo tiempo que el mensaje tarda en tránsito y procesamiento, más la desviación de los relojes), y 2) opcionalmente, ningún mensaje con la misma instantánea se recibió previamente del mismo originador [50] (pp. 399 - 400).

La desventaja de las instantáneas radica en que los protocolos que las usan requieren que los relojes estén sincronizados y asegurados contra modificaciones [36, 37]. También se requiere una tolerancia a la desviación propia de los relojes.

2.2.3 Métricas

Para medir la eficiencia de un protocolo de autenticación se utilizan parámetros precisos que sirven de comparación con otros protocolos u otras versiones del mismo protocolo, y en base a ellos poder decir si un protocolo es más o menos eficiente que otro. Las principales medidas de eficiencia para un protocolo de autenticación son las siguientes: el número total de *mensajes* intercambiados durante la ejecución del protocolo, y el número total de *rondas* del protocolo [10](pp. 4).

Para comprender mejor en qué consiste el número total de mensajes y el concepto de ronda, precisaremos lo que se entiende por mensaje. Un mensaje es una unidad de datos enviados por un participante en el protocolo a un único destino a la vez. Un mensaje puede contener encabezados para indicar su origen y destino. En algunas implementaciones, si el tamaño del mensaje es demasiado grande, el mensaje es fragmentado, al bajo nivel, en varios paquetes. En este caso, se considera a cada paquete como un único mensaje.

Número de Mensajes

Un mensaje que se transmite explícitamente, sin considerar retardos implícitos, como es el caso cuando es retransmitido por un *"gateway"*, cuenta como un nuevo mensaje enviado por una de las partes. Una difusión de un mensaje a n destinos, se considera como un mismo mensaje enviado a destinos diferentes y cuenta como n mensajes.

Para la transmisión de los mensajes, se hace la simplificación que la red está uniformemente conectada, de tal manera que los mensajes viajan a su destino siempre en una unidad de tiempo, sin importar dónde están el origen y el destino, y despreciando el tiempo de cómputo en cada uno de ellos. De esto se encargan las capas inferiores al nivel donde se ejecuta el protocolo.

Número de Rondas

Una ronda consiste de todos los mensajes que pueden ser enviados y recibidos en paralelo en una unidad de tiempo. Así, un participante puede, en una

ronda, simultáneamente enviar diferentes mensajes a diferentes destinos. Por tanto, múltiples participantes pueden enviar mensajes en una ronda. El número de rondas en un protocolo es el número total de unidades de tiempo desde el instante en que el originador envía el primer mensaje hasta el instante en que el último mensaje es recibido.

2.2.4 Notación

A lo largo de este trabajo, se utilizará la siguiente notación, que es una de las más comunes en la literatura sobre el tema:

A, B, G - Nombres para principales como: estaciones, centrales, clientes o servidores. Normalmente A funge como cliente y B como servidor.

$E_K(X)$ - Cifrado del mensaje en claro X, utilizando un algoritmo E de llave secreta K

$D_K(Y)$ - Descifrado del mensaje cifrado Y, utilizando un algoritmo D de llave secreta K

$E_{K_A^p}(X)$ - Cifrado del mensaje en claro X, utilizando un algoritmo E de llave pública y usando como llave, la llave pública de A, K_A^p

$D_{K_A^r}(Y)$ - Descifrado del mensaje cifrado Y, utilizando un algoritmo D de llave pública y usando como llave, la llave privada de A, K_A^r

$A \longrightarrow B : A, msg$ - El principal A envía al principal B su identidad, concatenada con el mensaje msg

P - El password: secreto involucrado entre las partes que desean identificarse, unilateral o mutuamente, y que sirve de base para la autenticación.

P_A - Password en claro del principal A

K_A - Llave secreta del principal A

K_A^p - Llave pública del principal A

K_A^r - Llave privada del principal A

k - Llave final de sesión, generada aleatoriamente por un principal y válida solamente para la sesión en que se generó.

k_i - Llave inicial de sesión, generada aleatoriamente por un principal. Esta notación se usa para indicar que no se trata de la llave final de sesión sino de alguna llave intermedia.

k_s - Llave de sesión generada por los principales A y B a partir de llaves de sesión k_A y k_B generadas por cada principal. Esta notación se usa para indicar que esta llave se genera a partir de llaves de sesión que ambos participantes generan.

N_A - Tipo de *núnico*. Elemento numérico generado aleatoriamente por el principal A, para usarse sólo una vez en cada sesión como elemento de unicidad.

C_A - Elemento numérico llamado *confundidor*, generado aleatoriamente por el principal A con el único objeto de confundir a un atacante, al incluirlo en el mensaje intercambiado con la otra parte e imposibilitar así redundancias que el atacante podría explotar en su favor.

T_A - Tipo especial de *núnico* generado por A llamado *instantánea*. Pieza de información reconocible pero no repetida, tal como el tiempo del reloj local, registrada con una presición mayor que el máximo permitido a la parte que actúa como cliente para el ajuste del reloj de la parte que actúa como servidor. Es generada, normalmente, por la parte que actúa como cliente.

Ch_A - Tipo de *núnico* conocido genéricamente como *reto, requerimiento, o quienvive*, usado como elemento de intercambio entre las partes durante el proceso de autenticación; generado, normalmente, de manera aleatoria por el principal A, que actúa como servidor.

f, f_1, f_2, g, h - Funciones hash unidireccionales

$Z \bigoplus W$ - Operación XOR bit a bit entre los vectores binarios Z y W

Δ - Ajuste máximo permisible del reloj local tanto del cliente como del servidor, en un esquema que utiliza instantáneas como elementos de unicidad

para garantizar la frescura de los mensajes

R_A, R_B - Exponentes aleatorios generados por A y B respectivamente, usados para el algoritmo de acuerdo de llave Diffie-Hellman [3]. El tamaño adecuado para estos exponentes es de 336 bits [19] (pp. 17)

$S_{K_r}(M)$ - Firma digital del mensaje M con llave privada K_r

$V_{K^p}(X, M)$ - Verificación de la firma X del mensaje M con la llave pública K^p

p - Número primo grande (alrededor de 1024 bits), adecuado para exponenciación (mod p) en el algoritmo de acuerdo de llave Diffie-Hellman. Este número se acuerda previamente entre las partes y ese acuerdo puede ser público

g - Número primo, menor que p anterior, que se utiliza como base para el algorimo de Diffie-Hellman. También puede acordarse públicamente entre las partes.

α, β - Base y módulo en notación equivalente a g y p anteriores, respectivamente, para el algoritmo Diffie-Hellman

q - Número entero que es factor primo grande de $p - 1$

G_x - Subgrupo [4] de Z_p^* de orden x, donde x es factor de $p - 1$

$F(P)$ - Función que convierte el password P a una base g adecuada para el algoritmo de Diffie-Hellman

[3] Sección A.11 del Apéndice A
[4] Sección A.10 del Apéndice A

2.3 Ataques a Protocolos

Un *ataque* consiste de cualquier acción que compromete la seguridad de la información. Al ente que realiza un ataque se le conoce como *atacante, intruso, adversario, oponente o enemigo.*

Cuando se estudia la seguridad de los protocolos criptográficos, se asume que los mecanismos criptográficos que utilizan, tales como algoritmos de cifrado y esquemas de firma digital, son seguros. Si no fuera así, entonces no habría esperanza de diseñar un protocolo seguro. Se supone también que el posible atacante no ataca directamente por criptoanálsis los mecanismos básicos, sino que intenta subvertir los objetivos del protocolo atacando la manera en la cual tales mecanismos son combinados, es decir, atacando al protocolo mismo [50] (pp. 495).

Algunas veces se hace distinción entre atacantes basándose en el tipo de información de que ellos disponen. Un atacante es *externo* cuando no tiene ningún conocimiento especial más allá del que se dispone monitoreando la ejecución del protocolo sobre líneas de comunicación abiertas. Un atacante es *interno* cuando tiene acceso a información adicional; por ejemplo llaves de sesión o información secreta parcial, obtenida por medios privilegiados, tales como acceso físico a recursos de cómputo, conspiración, etc. Un atacante es *interno por única vez* cuando obtiene información en un momento determinado y la usa en momentos subsecuentes; un atacante es *interno permanente* cuando tiene acceso continuo a información privilegiada [50] (pp. 496).

2.3.1 Estrategias de Ataques a Protocolos

El estudio de los ataques exitosos a protocolos que se creían seguros ha permitido aprender de los errores en el diseño de tales protocolos, entender los métodos y las estrategias generales de ataque, y formular principios y características para la corrección de fallas de seguridad y para el diseño de nuevos protocolos.

El atacante de un protocolo puede planear muchas estrategias con diferentes objetivos [50] (pp. 495 - 496). Estos objetivos pueden incluír, entre otros, los siguientes:

1. Deducir una llave de sesión usando información obtenida al monitorear la ejecución del protocolo

2. Participar secretamente en un protocolo iniciado por una de las partes legítimas con otra parte igual, e influír en el protocolo. Por ejemplo, alterando mensajes con el propósito de deducir una llave

3. Iniciar una o más ejecuciones del protocolo, posiblemente de modo simultáneo, y combinar los mensajes de una ejecución con los de otra con el fin de suplantar a una de las partes o llevar a cabo algún otro tipo de ataque

4. Engañar a una de las partes legítimas, refiriéndose a la identidad de la otra parte legítima con la cual comparte una llave

2.3.2 Ataques Pasivos y Activos

Hay varias formas de atacar un protocolo. El atacante puede monitorear y registrar alguna parte o toda la ejecución del protocolo. Esto se conoce como un *ataque pasivo* al protocolo, porque el atacante no lo afecta; lo que hace es observar y obtener información que podría utilizar después para montar un ataque activo. Un ataque pasivo sólo amenaza la confidencialidad de la información [50] (pp. 41).

En un *ataque activo*, un atacante puede intentar alterar el protocolo en su provecho. Puede tratar de hacerse pasar por una de las partes, introducir nuevos mensajes en el protocolo, borrar mensajes existentes, sustituir un mensaje por otro, devolver mensajes anteriores, interrumpir la línea de comunicación, o alterar información almacenada. Este tipo de ataque requiere una intervención activa por parte del atacante y amenaza la autenticación, la integridad y confidencialidad de la información.

Los ataques activos pueden tener diversos objetivos, tales como: obtener información, degradar el rendimiento del sistema, corromper la información almacenada, o tener acceso sin autorización a los recursos.

Un atacante que realiza un ataque activo, no necesariamente tiene que ser externo al sistema; puede ser un usuario legítimo del sistema, puede ser el administrador mismo; pueden ser varios atacantes trabajando juntos o

puede ser una de las partes involucradas en el protocolo, quien puede seguir toda o parte de la ejecución.

2.3.3 Ataques a Esquemas Basados en Password

Réplica de passwords

Este ataque se presenta cuando un atacante aprende un password observando al usuario cuando lo teclea. También se presenta cuando el password que el usuario teclea, o el resultado hash de él, se transmite en claro por la línea de comunicación entre el usuario y el sistema, y también cuando el password está temporalmente disponible en claro en la memoria durante el proceso de autenticación. Un atacante que esté monitoreando, puede registrar ese dato, permitiendo la suplantación subsecuente [50] (pp. 391).

Búsqueda Exhaustiva del password

Este es un ataque muy ingenuo: el atacante (aleatoria o sistemáticamente) prueba passwords, uno a la vez, sobre el autenticador actual, con la esperanza de hallar el password correcto. Este ataque se puede contrarrestar asegurándose que los passwords se escojan de espacios suficientemente grandes, limitando el número de intentos inválidos (en línea) permitidos dentro de periodos de tiempo fijos, y buscando que la transformación aplicada a los passwords, o el proceso de autenticación mismo, sean lentos [50] (pp. 391).

Este ataque puede hacerse fuera de línea. Por ejemplo, dado un archivo que contenga resultados hash de passwords, un atacante puede probar passwords, uno a la vez, aplicarle la función hash, y comparar el resultado hash de cada password en el archivo de passwords cifrados. Esto es teóricamente posible ya que tanto la función hash como el password probado se conocen. Este ataque puede evitarse manteniendo en secreto los detalles de la función hash o el archivo mismo, pero no se considera prudente basar la seguridad de un sistema en la suposición de que tales detalles permanecerán en secreto para siempre.

La viabilidad del ataque depende del número de passwords que necesiten ser

verificados antes de que ocurra una coincidencia (la cual depende del número de posibles passwords), y del tiempo requerido para probar cada uno de ellos. Este último depende de la función hash que se utilice, de su implementación, del tiempo de ejecución para cada instrucción del procesador, y del número de procesadores disponible (la búsqueda exhaustiva es paralelizable). El tiempo requerido para comparar el resultado hash de cada password probado en el archivo de passwords, es típicamente despreciable [50] (pp. 391).

Lo anterior involucra computaciones típicamente grandes, las cuales no requieren interactuar con el autenticador actual sino hasta en una etapa final.

Conjetura de passwords y Diccionario

Para mejorar la probabilidad de éxito de una búsqueda exhaustiva, en lugar de buscar sobre el espacio de todos los posibles passwords, un atacante puede barrer ese espacio en orden decreciente de acuerdo a la probabilidad de ocurrencia de los passwords.

Idealmente, cualquier cadena arbitraria de n caracteres tiene la misma probabilidad de ocurrir que los passwords seleccionados por el usuario. Algunos estudios indican que una gran fracción de los passwords seleccionados por el usuario se encuentran en diccionarios típicos de sólo 150000 palabras, mientras que un gran diccionario de 250000 palabras representa solamente una pequeña fracción de todos los posibles passwords de n caracteres [50] (pp. 392).

Cualquier lista de palabras puede ser usada por un atacante, el cual prueba como passwords todas las palabras disponibles en la lista, usando el así llamado ataque *diccionario* por estar basado precisamente en un diccionario de posibles passwords. Además de los diccionarios tradicionales, se pueden usar diccionarios en lenguas extranjeras, o sobre tópicos especializados tales como música, películas, etc. [50] (pp. 392).

Existen dos maneras en las que un atacante puede usar un diccionario. La primera se conoce como *ataque por diccionario fuera de línea*, y consiste en que el atacante registra mensajes intercambiados entre participantes legítimos de un protocolo de autenticación. Después, realiza pruebas con candidatos a passwords extraídos de su diccionario y observa si el resultado

es consistente con los mensajes registrados. Si encuentra que un candidato a password es consistente con sus registros, entonces concluye que ese candidato es el password del usuario. Esto si el atacante no tiene acceso al archivo de passwords que, como en el caso del sistema operativo $UNIX$, puede contener los passwords cifrados de alguna manera y, puede ser público para usuarios legítimos. Si el atacante es capaz, por algún medio, de obtener el archivo de passwords, este ataque es aún más fácil ya que puede, con sus propios recursos de cómputo en otro momento, probar los passwords de sus diccionarios y comparar los resultados con las entradas correspondientes en el archivo. Esta forma de ataque asume que el atacante conoce la forma en que los passwords son cifrados.

La otra forma de este ataque se conoce como *ataque por diccionario en línea*, y consiste en que el atacante repetidamente selecciona un password de su diccionario e intenta utilizarlo para, en ese momento, suplantar al usuario. Si la suplantación falla, el atacante descarta ese password de su diccionario e intenta de nuevo con otro diferente. Las maneras estándares de prevenir esta forma del ataque diccionario es limitar el número de ejecuciones fallidas del protocolo de autenticación que a un usuario le está permitido hacer antes de expirar el password. También, reducir el número de intentos de *"login"* permitidos al usuario antes de, por ejemplo, bloquear su terminal [61] (pp. 3 - 4).

Por razones de eficiencia, un atacante también puede crear y guardar en disco o cinta, diccionarios con resultados hash, o diccionarios de passwords con alta probabilidad de ocurrir. El atacante puede entonces coleccionar archivos que guarden resultados hash de passwords, ordenarlos y compararlos con las entradas en el diccionario. Los ataques tipo diccionario generalmente no son útiles para hallar un password particular de un usuario, sino para hallar múltiples passwords en la mayoría de sistemas [50] (pp. 392).

Los ataques por diccionario son más efectivos cuando un gran número de intentos por adivinar el password, pueden hacerse automáticamente y cada intento puede ser verificado si fue correcto o no, sin provocar alguna alarma [1] en el sistema. Este ataque por diccionario es el que la mayoría de protocolos criptográficos basados en passwords tratan de evitar.

Estos ataques están frecuentemente asociados con sistemas de passwords almacenados, tal como sucede en el sistema operativo $UNIX$, donde lo que se guarda en el servidor es un archivo conteniendo, entre otras cosas, el

resultado de aplicar una función hash al password [1] (pp. 1). Pero también pueden aplicarse a sistemas de autenticación en red que utilizan el modelo de Needham y Schroeder [26] de distribución de llave temporal, donde un servidor conoce una llave personal para cada usuario. Este riesgo surge porque al hacer que tales sistemas sean aceptables para el usuario, las llaves personales de los usuarios pueden haber sido derivadas de passwords que el usuario escoge [6] (pp. 1).

Por ejemplo, el sistema de autenticación de Kerberos [4] es vulnerable a ataques por diccionario, ya que el servidor de llaves cifra su respuesta inicial usando una llave derivada del password del usuario. Un atacante puede registrar tal mensaje e intentar descifrarlo usando llaves derivadas de una serie de conjeturas para el password. El atacante puede fácilmente determinar si una conjetura es o no correcta debido a que, para una conjetura correcta, el resultado del descifrado del mensaje producirá datos reconocibles, tales como la hora del día o el nombre de un servicio de red. Esta prueba con numerosos passwords puede hacerse fuera de línea [1] (pp. 1 - 2).

2.3.4 Ataques a Protocolos de Autenticación

Ataque por Suplantación

Este ataque consiste en un engaño por medio del cual un atacante suplanta a un participante legítimo del protocolo. Las amenazas potenciales a los protocolos de autenticación incluyen suplantación o engaño por cualquiera de los siguientes ataques: *réplica*, *combinación*, *sesión paralela* (también llamado *reflexión*) y *retardo forzado*. La suplantación también es trivialmente posible si un atacante es capaz de descubrir alguna información relacionada con la llave permanente (secreta o privada) de un participante, por ejemplo, usando un ataque por *texto escogido*. Estos ataques [50] (pp. 417 - 418) se describirán enseguida.

Ataque por Réplica

Este ataque consiste de una suplantación o algún otro engaño que un atacante hace al usar información proveniente de una ejecución pasada del

protocolo sobre el mismo o distinto autenticador, y hacerla pasar por información auténtica en otra ejecución. Esta es una de las razones por la que se busca que los protocolos de autenticación con intercambio de llaves, establezcan una llave de sesión temporal, además del password que ya comparten, para poder cifrar los intercambios con esa llave temporal durante esa sesión. Por ejemplo, para archivos almacenados el caso análogo de este ataque es un ataque por *restauración*, por medio del cual un archivo es reemplazado por una versión anterior del mismo archivo [50] (pp. 417).

Ataque por Texto escogido

Este es un ataque[5] típico sobre un protocolo tipo quienvive-respuesta, en el cual el atacante selecciona estratégicamente quienvives intentando extraer información acerca de la llave fija de la parte que se identifica. Este ataque es algunas veces referido por usar a esa parte como un oráculo (Ver Ataque por *Combinación* en esta misma Sección). Este ataque puede involucrar texto en claro escogido (Subsección A.3.1 del Apéndice A) si la parte que se autentica requiere firmar, cifrar o autenticar el quienvive; o puede involucrar texto cifrado escogido si el requerimiento es para descifrar un quienvive [50] (pp. 417).

Ataque por Combinación (*"interleaving"*)

Este ataque consiste de una suplantación u otro engaño que involucre la combinación selectiva de información perteneciente a una o más ejecuciones previas o simultáneamente en proceso (sesiones paralelas) del protocolo, incluyendo la posible originación de una o más ejecuciones del protocolo por parte del propio atacante [25, 43].

El término *combinación* agrupa a un conjunto de ataques, entre los cuales están los siguientes [43] (pp. 6 - 9):

1. *Texto en claro conocido*

[5] Aunque este es un ataque por criptoanálisis, se menciona en este contexto por ser muy utilizado para atacar protocolos de autenticación

2. *Texto cifrado escogido*

3. *Oráculo*

4. *Sesión Paralela o Reflexión*

Para ilustrar claramente estos ataques, supongamos que se tiene el siguiente protocolo elemental ilustrado en [43] (pp. 5):

1. A genera el núnico N_1, lo cifra usando como llave el password P, y lo envía a B

   $A \longrightarrow B : E_P(N_1)$

2. B descifra $E_P(N_1)$ porque comparte el password P con A, genera su propio núnico N_2 y lo envía a A junto con N_1

   $B \longrightarrow A : N_1, E_P(N_2)$

3. A descifra $E_P(N_2)$, conoce N_2, y lo envía a B

   $A \longrightarrow B : N_2$

Un ataque por texto en claro conocido es posible en este protocolo porque el mismo núnico que viaja cifrado, posteriormente viaja en claro. El atacante puede juntar parejas de mensajes (texto en claro, texto cifrado) para criptoanalizarlas después. Este ataque es pasivo.

También es posible atacar este protocolo por texto cifrado escogido porque el atacante puede simular ser A o B, enviar a la otra parte un texto cifrado que él mismo selecciona y esperar que la otra parte le conteste con el texto descifrado. El atacante no conoce la llave, pero puede acumular parejas de mensajes (texto en claro, texto cifrado) de las cuales él mismo escogió el texto cifrado (o el texto en claro si el núnico se cifra en lugar de descifrarse). Este ataque es activo.

Un ataque por *oráculo* al mismo protocolo anterior, sería el siguiente [43] (pp. 7):

1. El atacante I previamente ha registrado los mensajes intercambiados en una ejecución legal del protocolo entre A y B; en particular tiene

registrado $E_P(N_1)$. Ahora, para realizar este ataque simula ser A, e inicia una sesión con B, enviándole

$I \longrightarrow B : E_P(N_1)$

2. B cree que quien le hace el envío es A y responde enviando

   $B \longrightarrow I : N_1, E_P(N_2)$

3. Como I no conoce P, no puede descifrar $E_P(N_2)$. En este momento, I simula ser B, e inicia una sesión con A enviándole $E_P(N_2)$ del intercambio 2 de la sesión con B

   $I \longrightarrow A : E_P(N_2)$

4. A cree que quien le hace el envío es B y responde enviando

   $A \longrightarrow I : N_2, E_P(N_3)$

5. I recibe este envío de A y de esta forma conoce N_2, sin descifrar $E_P(N_2)$. Toma N_2 y se lo envía a B en la sesión que mantiene abierta con él

   $I \longrightarrow B : N_2$

6. I abandona la sesión con A

En este ataque, I rompe el protocolo sin romper la llave: no puede descifrar $E_P(N_2)$ pero puede atacar por texto cifrado escogido a A, al usarlo como oráculo para saber N_2.

Este ataque muestra una debilidad fundamental del protocolo: los mensajes criptográficos en cada intercambio del protocolo deben ser distintos en el sentido que no debe ser posible a un atacante usar un mensaje que aparezca en algún intercambio para derivar, reconstruir o falsificar mensajes requeridos en otro intercambio [43](pp. 7).

Para superar este problema se puede pensar en una mejora del protocolo anterior definido por el ISO SC27 ISOSC27, e ilustrado en [43] (pp. 8). La mejora consiste en que ahora B debe ser capaz de cifrar un núnico y A debe ser capaz de descifrarlo. De este modo, un atacante no puede usar a una parte como oráculo contra la otra. El protocolo mejorado es el siguiente:

1. A genera el núnico N_1, y lo envía a B

$A \longrightarrow B : N_1$

2. B recibe N_1, lo cifra usando el password P como llave. Genera su propio núnico N_2, lo cifra usando el password P como llave, y envía los dos mensajes cifrados a A

   $B \longrightarrow A : E_P(N_1), E_P(N_2)$

3. A descifra $E_P(N_1)$ y $E_P(N_2)$, conoce N_1 y N_2, verifica que N_1 sea el que envió y envía N_2 a B

   $A \longrightarrow B : N_2$

Este protocolo mejorado puede aún ser atacado por medio del ataque conocido como *sesión paralela*, también llamado por *reflexión* debido a que involucra el envío de información desde una ejecución en curso, de regreso al originador de tal información. Este es un ataque pasivo que se ilustra en [43] (pp. 8 - 9) y consiste en lo siguiente:

1. El intruso I intercepta una ejecución legal del protocolo entre A y B iniciada por A. Captura N_1, bloquea completamente la comunicación hacia B dejándolo totalmente fuera de escena, simula ser B y hace que A realice un oráculo contra sí mismo. Habiendo hecho lo anterior, inicia una ejecución paralela con A

   El primer mensaje de la sesión legal, que I captura, es

   $A \longrightarrow B : N_1$

   Puesto que I no puede replicar con $E_P(N_1)$ porque no conoce P, se hace pasar por B y le envía a A, en la ejecución paralela, el mismo N_1 que capturó de la ejecución legal

   $I \longrightarrow A : N_1$

2. A cree que quien le hace el envío es B y responde, de acuerdo al protocolo original, con el segundo mensaje de la sesión paralela. Este mensaje contiene $E_P(N_1)$, justamente lo que I necesita para completar la primera autenticación. También en este mensaje A envía su propio núnico cifrado $E_P(N_2)$

   $A \longrightarrow I : E_P(N_1), E_P(N_2)$

   En este punto, como I no conoce P, no puede descifrar $E_P(N_2)$ por lo que debe interceptar el tercer mensaje, (N_2), de la primera ejecución

entre A y B. Para esto, I bloquea esta ejecución paralela, reinicia la comunicación de la ejecución legal con A y haciéndose pasar otra vez, en esta ejecución, por B ante A, le envía el segundo mensaje de la ejecución legal

$I \longrightarrow A : E_P(N_1), E_P(N_2)$

3. A cree que B le contesta su primer mensaje de la ejecución legal y completa esta primera autenticación enviando N_2, que es otra vez exactamente lo que I necesita para completar la segunda autenticación.

   El tercer mensaje de la sesión legal es

   $A \longrightarrow I : N_2$

   De este modo I se ha hecho pasar por B en ambas ejecuciones: la original y la paralela. De hecho, I puede abandonar en este punto una de las ejecuciones

   El tercer mensaje de la sesión paralela es

   $I \longrightarrow A : N_2$

Este ataque ilustra otra debilidad fundamental de muchos protocolos simples de autenticación: la expresión criptográfica usada en el segundo mensaje debe ser asimétrica (es decir, dependiente de la dirección) de tal forma que su valor en una ejecución del protocolo iniciada por A no pueda ser usada en una ejecución del protocolo iniciada por B [43] (pp. 9).

Muchos protocolos, en diferentes capas de distintas arquitecturas de red, no permiten el establecimiento simultáneo de sesiones paralelas múltiples. Sin embargo, también existen muchos ambientes de red donde tales sesiones paralelas son legales por diseño. En estos casos se puede pedir que los implementadores verifiquen que el primer núnico recibido en una ejecución no sea una réplica de un núnico de otra sesión. No obstante, dejar la seguridad de la ejecución a los implementadores, sin garantizarla desde el diseño del protocolo, es una mala práctica. Por otro lado, en implementaciones típicas, los parámetros de ejecución se instancian de manera independiente para cada ejecución, de tal forma que no es posible en una ejecución verificar parámetros de otra [43] (pp. 8).

Ataque por Hombre Enmedio

Este es un tipo de ataque activo que se ilustra muy claramente en [38] (pp. 48 - 49) suponiendo un protocolo de autenticación que utiliza criptografiía de llave pública para el intercambio de las llaves.

El atacante se coloca enmedio de la línea de comunicación entre las partes A y B. Cuando A envía su llave pública a B, el atacante la intercepta, genera una nueva llave y la envía a B. B entonces recibe una llave que supone es la de A. B envía su llave pública a A por la misma línea. El atacante la intercepta y la reemplaza por una de su propiedad.

Ahora, A y B inician el intercambio de mensajes. El atacante intercepta un mensaje desde A, cifrado con la llave que A cree que es de B pero que en realidad es del atacante, el cual puede descifrar el mensaje. El atacante cifra este mensaje en claro con la llave pública real de B y envía el texto cifrado a B. B recibe este mensaje como válido, pues está cifrado con su llave pública, y asume que el enviador es A.

A y B nunca notarán que alguien está leyendo sus mensajes. Todavía peor: el atacante no solo puede leer la conversación cifrada, sino que también puede modificarla o cambiarla completamente. Este ataque es duro y muy difícil de defender.

Ataque por Negación de Servicio

Este ataque puede hacerse en diferentes ámbitos y niveles. Típicamente un ataque por negación de servicio evita o inhibe el uso o manejo normal de las facilidades de comunicación, o de algún recurso de cómputo. Este ataque puede tener objetivos específicos. Por ejemplo, el atacante puede suprimir todos los mensajes dirigidos a un destino particular. Otra forma del ataque es el bloqueo de toda una red, ya sea desabilitándola, o bien sobrecargándola con mensajes para degradar su rendimiento [45] (pp. 10).

A nivel de aplicación, un ataque de este tipo puede ser montado contra un usuario a través de un programa que se ejecute en el ambiente de ese usuario y que, por ejemplo, llene su monitor con pantallas de información

hasta agotar la memoria de la máqina y dejar así imposibilitado al usuario para seguir utilizando sus recursos de cómputo.

En el ámbito de los protocolos de autenticación, el atacante intenta hacer que el protocolo falle y no proporcione el servicio o función para el que fue diseñado. Este ataque es muy difícil de defender o evitar, pero a diferencia de ataque por hombre enmedio, este es muy fácil de detectar y generalmente no compromete la seguridad de la información.

Ataque por Texto Verificable

El ataque por *texto verificable* es muy similar al ataque por texto en claro conocido, pero *más general*. Si un mensaje contiene información que es reconocible cuando se descifra, haya sido o no predecible con anticipación, entonces se describe el texto como verificable. Un ejemplo de mensaje que no contiene texto en claro conocido, pero contiene texto verificable es cuando se tiene un protocolo de autenticación en el cual A intenta convencer a B que conoce una llave secreta previamente acordada. A selecciona una cadena aleatoria, por ejemplo de 64 bits, y envía a B un mensaje cifrado con la llave secreta; el mensaje cifrado contiene esa cadena aleatoria concatenada a su complemento a unos. B verifica el mensaje de A descifrando el mensaje con la llave secreta y sumando los dos números para ver si el complemento a unos del resultado da cero. Este mensaje, a pesar de no contener texto en claro conocido, contiene texto verificable porque un atacante familiarizado con el protocolo puede atacarlo intentando descifrar el mensaje conjeturando llaves y observando aquellas llaves para las cuales el complemento a unos del resultado de la suma de los números descifrados da cero. Si más de una lleve satisface esta condición, el atacante puede requerir monitorear posteriores intercambios, pero tendrá suficiente información para verificar sus conjeturas sobre la llave [6] (pp. 3).

El texto verificable puede ser texto en claro, texto cifrado, o puede ser el resultado de un encadenamiento de cálculos que pueden o no involucrar cifrado o descifrado. Para descubrir un password débil, un atacante conjetura posibles passwords y, junto con su conocimiento sobre el sistema, realiza cálculos para ver si puede descubrir algo que ya sabe o que puede reconocer [1] (pp. 5). Por esta razón, a este tipo de ataque también se le conoce como *texto en claro reconocible*.

En este contexto, se usa la palabra reconocer en un sentido general: por ejemplo, si se cifra y envía un mensaje con texto verificable, y en algún otro mensaje un atacante intercepta algo que puede ser la llave de cifrado, el atacante puede verificar la llave intentando descifrar el texto cifrado con esa llave y observando el resultado. Tal llave es reconocible y por tanto verificable. Es por esta razón que se debe ser cuidadoso al cifrar llaves, aún llaves fuertes, usando llaves débiles tales como los passwords porque se pueden probar rápidamente y con los resultados de los descifrados probar a su vez descifrar un texto verificable [6] (pp. 3).

El aspecto más inquietante del texto verificable desde el punto de vista del diseño del protocolo, es que un atacante puede verificar la correctez de una conjetura, tal como la llave de cifrado, sin ninguna interacción con alguna parte del sistema que pudiera registrar y advertir intentos fallidos. Este aspecto hace difícil al sistema reconocer cuando alguien está realizando tal ataque [6] (pp. 3).

Si se usa un criptosistema de llave pública, otra forma de ataque por texto verificable es permitir al atacante determinar el texto en claro sin descubrir la llave. El intruso registra una o más transacciones usando una llave pública conocida. Para cada posible valor del texto en claro, el intruso calcula el correspondiente texto cifrado y lo compara con el texto cifrado que registró previamente. Si hay coincidencia, significa que el texto en claro ha sido encontrado [6] (pp. 3).

Es posible construir una serie de mensajes, ninguno de los cuales es vulnerable en sí mismo a este tipo de ataque, pero juntos pueden contener texto verificable. Por ejemplo, un protocolo puede usar el password para cifrar un núnico; el atacante no tiene modo de saber si ha adivinado correctamente el núnico ya que este es aleatorio. No obstante, si el complemento a unos de ese núnico se cifra más tarde con el mismo password en otro paso del protocolo, el atacante puede intentar descifrar ambos mensajes con un ataque sobre passwords y verificar si los dos mensajes en claro resultantes son complemento uno del otro y así verificar su conjetura [6] (pp. 3).

Esta forma de ataque puede evitarse introduciendo en el mensaje un número aleatorio, *confundidor*, distinto a un núnico en que no tiene otro propósito que confundir un ataque de este tipo. El valor de un confundidor puede ser ignorado por el receptor del mensaje en el cual aparece [6] (pp. 3).

2.4 Advertencias acerca de la Caracterización de Ataques

En el área de seguridad en cómputo, y en la de protocolos de autenticación e intercambio de llaves en particular, caracterizar todos los tipos de ataques de manera completa y útil, no es trivial. Esto se debe a que continuamente se descubren nuevos tipos de ataques, a que la relación entre los ataques y los efectos combinados de ellos no es clara, y a que no hay consenso respecto a si todas las clases de ataques conocidos tienen soluciones satisfactorias [9] (pp. 2).

Por tanto, cuando se dice que un protocolo es seguro, tal afirmación se hace contra un conjunto de supuestos ataques y depende frecuentemente de suposiciones acerca de la naturaleza de los protocolos, o de los mensajes que pueden ser violados en presencia de un tipo de ataque no previsto. El no hallar vulnerabilidades no significa que el protocolo sea seguro, sino que ciertas líneas de ataques son menos probables que sucedan [9] (pp. 1).

Capítulo 3

Alternativas y Soluciones

En las secciones 1.3.1 y 1.3.2 del capítulo 1 de este trabajo se mencionan algunas alternativas posibles para esquemas convencionales de password. Ninguna de las alternativas mencionadas resuelve el problema de autenticación segura basada en passwords y la elección de passwords débiles.

En lugar de obligar a los usuarios a escoger passwords criptográficamente fuertes, complicados y difíciles de recordar, lo ideal es tener soluciones o protocolos que mantengan, al mismo tiempo, la comodidad del usuario y un alto nivel de seguridad. Una de las ideas básicas es asegurar que los datos disponibles al atacante sean lo suficientemente impredecibles para prevenir una verificación fuera de línea acerca de si un password que se prueba es válido o no [1] (pp. 1).

En este capítulo, se hace una revisión de las alternativas más importantes que existen hasta la fecha, empezando, por su trascendencia histórica y actual, con el esquema de autenticación en UNIX, su desarrollo histórico, su problemática de seguridad, y sus alternativas actuales. Después, se ilustra la evolución de alternativas y soluciones al problema de autenticación basada en passwords, hasta llegar al estado que guardan actualmente tanto las implementaciones en uso como las soluciones existentes y direcciones de investigación dentro de lo que se conocen como soluciones, métodos, o protocolos fuertes.

3.1 Sistemas UNIX

Los sistemas operativos *UNIX* son, en la actualidad, los sistemas más ampliamente conocidos y utilizados; basan su esquema de autenticación en *passwords seleccionados* por los usuarios. Este esquema de autenticación ha sufrido cambios y transformaciones a lo largo de la historia de *UNIX* como sistema operativo. Por su importancia actual y por sus posibilidades futuras, en esta sección se hace un estudio del esquema de autenticación de *UNIX*, se muestra su desarrollo histórico, y se describe su problemática de seguridad actual y sus distintas alternativas de solución.

3.1.1 Autenticación en UNIX

El proceso de autenticación en los sistemas *UNIX*, se basa en un esquema de passwords que utiliza un archivo de acceso público llamado */etc/passwd*. Este archivo contiene un campo que guarda el resultado de aplicar una función unidireccional a cada password P de usuario. En este contexto se entiende por función unidireccional una función que es fácil de calcular en un sentido pero computacionalmente infactible de calcular en el otro. En el sentido más general esta es una de las propiedades de las funciones llamadas hash y hash unidireccionales (para detalles sobre esto, ver Subsección A.3.3 del Apéndice A). Es esta la razón por la cual algunas veces se dice que el archivo de passwords de *UNIX* guarda resultados hash de los passwords de usuarios. Este resultado se genera de la siguiente manera. Cada password P, se usa como llave para cifrar un texto en claro conocido (64 bits de ceros) utilizando como algoritmo de cifrado una modificación menor del algoritmo de llave secreta *DES*(*"Data Encryption Standard"*) [14]. La seguridad de esta técnica se basa en la propiedad de *DES* de resisitir ataques por texto en claro conocido, lo cual significa que dado el texto en claro y su correspondiente texto cifrado, resulta difícil hallar la llave de cifrado (Subsección A.3.1 del Apéndice A).

Cada vez que el usuario desea autenticarse ante el sistema, este le pide teclear su identidad o *"login"* para enseguida pedirle la verificación de esa identidad, o sea el password. El sistema aplica la función unidireccional al password tecleado y compara el resultado que obtiene con el que guarda en el archivo de passwords. Si hay coincidencia, da la autenticación por buena

y concede el acceso; si no, lo rechaza.

El detalle del proceso de cifrado usando el password como llave es el siguiente. El password es truncado a sus primeros 8 caracteres *ASCII*, cada uno de los cuales proporciona 7 bits para la llave *DES* de 56 bits (si el password es menor a 8 caracteres, se rellena con ceros). El password se usa como llave para cifrar 64 bits de ceros; las salidas producidas se usan como entradas sucesivas para iterar 25 veces el *DES*. El resultado final de 64 bits es reempacado para convertirlo en 11 caracteres imprimibles (esto es necesario porque la salida de 64 bits más los 12 bits de sal producen 76 bits y 11 caracteres *ASCII* requieren 77 bits) [50] (pp. 393 - 394). El método no estándar de "salar" el password se usa en *UNIX* para intentar simultáneamente complicar un ataque por diccionario y evitar que el atacante utilice implementaciones comerciales en hardware del *DES*.

El valor aleatorio conocido como sal lo asocia el sistema a cada password de usuario. Este valor consta de 12 bits (tomados del reloj del sistema al momento de la creación del password) y se utilizan para alterar la función de expansión estándar E del *DES* [50] (pp. 250 - 259), proporcionando de este modo una entre 4096 (2^{12}) posibles variaciones [50] (pp. 393 - 394).

La función de expansión E crea un bloque de 48 bits; inmediatamente después, los bits de la sal colectivamente determinan una de las 4096 permutaciones. Cada bit es asociado con una pareja predeterminada del bloque de 48 bits. Por ejemplo, el bit 1 de la sal se asocia con los bits 1 y 25 del bloque, el bit 2 de la sal con los bits 2 y 26 del bloque, etc. Si el bit de sal que se asocia a la pareja del bloque de 48 bits es 1, los bits del bloque se intercambian; si no, no [50] (pp. 393 - 394).

El resultado hash del password y la sal se codifican a caracteres imprimibles y ambos se guardan en el archivo de passwords del sistema. La seguridad de cualquier password particular de usuario no se modifica al usar la sal, pero un ataque por diccionario requiere ahora 4096 variaciones de cada password que se prueba. Del mismo modo, debido a que la función de expansión E del *DES* es dependiente ahora de la sal, los chips estándares que implementan *DES* no pueden usarse para implementar el algoritmo de passwords de *UNIX*. Un atacante que quiera ganar velocidad implementando el algoritmo en hardware, no puede usar chips comerciales; tiene que construir un chip especial, lo cual pretende desalentar a atacantes con recursos modestos [50] (pp. 393 - 394).

El punto aquí es asegurarse que el atacante tiene que cifrar cada password en su diccionario cada vez que intente romper el password de otra persona, y no basta hacer una precomputación masiva de todos los posibles passwords [38] (pp. 52 - 53).

La sal protege contra ataques diccionario generales sobre un archivo de passwords porque para cada uno de ellos se tienen que probar 4096 posibilidades; pero no protege contra un ataque concertado sobre un único password porque para un password en especial las 4096 posibilidades son computacionalmente factibles. También la sal protege a usuarios que tienen el mismo password en múltiples máquinas porque el resultado hash almacenado en el archivo de passwords de cada máquina no necesariamente es el mismo para el mismo password. No obstante, la sal no hace que la elección de passwords débiles sea mejor [50] (pp. 24 - 25).

Un ataque por diccionario sigue el mismo procedimiento que la verificación legítima de un password. El atacante prueba un password P' calculando $h(P', s)$, donde s es la sal y $h()$ es la función unidireccional, y comparando el resultado con el valor almacenado $h(P, s)$. Si hay coincidencia entonces asume que $P = P'$. Si no coincide, el atacante prueba con el siguiente candidato P' de una lista (diccionario) de posibles passwords [1] (pp. 3).

Por la importancia actual de los sistemas $UNIX$, a continuación se ofrece un panorama histórico de lo que ha sido su problemática de seguridad desde sus primeras versiones hasta la actualidad, analizando sus ataques por diccionario en términos de épocas, tiempo, recursos computacionales y tamaño de los diccionarios utilizados.

3.1.2 Panorama Histórico

Los primeros sistemas $UNIX$ usaron un algoritmo de cifrado que simulaba una máquina de rotor *M-209* [38] (pp. 12 - 13). Una desventaja de este algoritmo era su rapidez: ejecutándose sobre una máquina *PDP-11/70*, cada operación de cifrado tomaba aproximadamente 1.25 ms, de tal forma que se podían verificar alrededor de 800 passwords/seg. Usando un diccionario de 250000 palabras, un posible atacante podía comparar sus cifrados con todos los guardados en el archivo de passwords en poco más de 5 min. [13] (pp. 1).

En versiones posteriores (después de 1976), se usó el criptosistema de llave secreta *DES* [14] para cifrar una constante, utilizando el password como llave. Iterando 25 veces el *DES*, se obtiene una cadena de 11 caracteres a los cuales se les agrega la sal de 2 caracteres para hacer un total de 13 caracteres que se almacenan en el archivo */etc/passwd* (para detalles, ver la Subsección 3.1.1 de este mismo capítulo). Este método presentaba la ventaja de ser lento: ejecutándose sobre una máquina *microVAX-II*, la cual era sustancialmente más rápida que la *PDP-11/70*, una operación de cifrado tardaba sobre el orden de 280 ms, de tal forma que un atacante podía verificar 3.6 cifrados/seg. Verificar el mismo diccionario anterior, tomaba alrededor de 19 hrs. Verificar los passwords de un sistema con 50 usuarios podía tomar sobre 40 días sin ninguna garantía de éxito [13] (pp. 1).

A principios de los 90's, el creciente desarrollo en hardware y software hizo que la situación cambiara de nuevo, sobre todo en aspectos tan fundamentales como los siguientes [13] (pp. 2):

1. La velocidad de los procesadores había crecido de 25 a 40 veces desde 1976 y la interconexión de máquinas por medio de redes hacía posible tener una mayor capacidad de cómputo para reducir el trabajo.

2. Se habían desarrollado nuevas implementaciones del *DES* [14, 31] que tomaban menos de 1 ms en cifrar el password y compararlo con el que se guarda en el archivo de passwords. Ejecutándose sobre una estación de trabajo, el mismo diccionario de antes se podía cifrar y comparar en menos de 5 min. Interconectando estaciones, el tiempo requerido para cifrar esas palabras del diccionario contra los 4096 posibles variaciones que proporciona la sal podía ser menor a 1 hr.

También, a principios de los 90's se obtuvo el siguiente resultado importante: usando un diccionario de 62727 palabras (sin considerar varias permutaciones) se rompieron 21% de 15000 passwords usando aproximadamente una semana de tiempo de procesador. El primer 2.7% se encontró en 15 min. [13] (pp. 3).

Sin embargo, el viejo problema de la educación de los usuarios para elegir passwords criptográficamente fuertes seguía, y sigue, siendo el mismo.

3.1.3 Problemática Actual y Alternativas

En la actualidad, el problema de seguridad en *UNIX* para la autenticación de usuarios sigue siendo provocado por la elección de passwords criptográficamente débiles y por el hecho de que el archivo de passwords es público, y aunque lo que aparece en ese archivo es el resultado de aplicar una función unidireccional a los passwords, son vulnerables a ataques tipo diccionario desde dentro o fuera del sistema [13] (pp. 2).

Varias alternativas han sido propuestas e implementadas para superar los problemas de seguridad en el caso específico de *UNIX*.

Una de estas alternativas es que el administrador del sistema verifique o ataque su propio archivo de passwords y advierta a los usuarios que encuentre con passwords débiles. Esta alternativa, no obstante, tiene los siguientes inconvenientes [13] (pp. 2):

1. Mientras se verifica, el archivo de passwords sigue expuesto
2. La prueba consume tiempo y sólo reporta passwords capaces de romperse
3. Los passwords que caen fuera del caso específico, se pueden romper con diccionarios más sofisticados

Otra alternativa, actualmente también disponible para los sistemas *UNIX* de *SUN*, es hacer ilegible el archivo de passwords o el campo correspondiente al resultado hash del password. Esta opción, tiene los siguientes inconvenientes y limitaciones [13] (pp. 2):

1. El campo correspondiente al resltado hash debe guardarse en otro archivo (*/etc/shadow*) que también puede comprometerse o ser intervenido por atacantes internos que puedan violar los permisos de acceso
2. Es una alternativa particular para ciertos sabores de *UNIX* (los de *SUN*) que requiere instalar software adicional

Otra alternativa viable y actualmente en uso, es la implementación de un verificador de passwords *proactivo*, que permite a los usuarios cambiar su

password y verificar, "a priori", si el nuevo password es seguro [13] (pp. 2). Esta opción requiere instalar nuevo software y hacer cómputo adicional al proceso de autenticación y, de alguna manera, obliga a los usuarios a escoger passwords criptográficamente fuertes.

Las alternativas anteriores, adolecen, sin embargo, de un problema común no resuelto por ninguna de ellas: el password, cualquiera que sea, viaja en claro por la red, expuesto a cualquier intruso armado de un buen programa tipo *"sniffer"* que puede capturar paquetes. Desde luego, este problema se restringe a la seguridad sobre un segmento de una red y no a su interconectividad con otras redes, sobre todo si se trata de grandes redes; esto reduce y acota el riesgo, pero no lo evita.

De esta manera, la vulnerabilidad de los passwords en $UNIX$, para un atacante, se reduce a efectuar los siguientes pasos [13] (pp. 3):

1. Adquirir una copia del archivo */etc/passwd*, ya sea por medio de un enlace *uucp* no protegido, hoyos de seguridad en el programa *sendmail*, o por *ftp* o *tftp*
2. Aplicar el algoritmo de cifrado de passwords a una colección de palabras (diccionarios), más alguna permutación sobre *"logins"* y nombres de usuarios, y comparar los resultados con las entradas del archivo */etc/passwd* o equivalente.
3. Si un atacante se instala con un programa *"sniffer"* en un segmento específico de la red, es capaz de capturar paquetes conteniendo *"logins"* y passwords que viajan en claro.

Como casos concretos de estudios recientes realizados en sistemas $UNIX$ sobre la vulnerabilidad de passwords, están los siguientes:

Usando el programa de prueba de passwords llamado *Crack v4.1 ("ultra fast crypt")* y un diccionario de 250000 palabras de 6 letras minúsculas, sobre una computadora *SUN ELC* (20 $MIPS$ comparable a una PC moderna), se obtuvieron los siguientes resultados [15] (pp. 4):

1. **Número total de cuentas :** 521

2. **Número de passwords adivinados :** 58
3. **Porcentaje sobre total de cuentas:** 11.1%
4. **Tiempo Total :** 59:13 (Tiempo real)

Probando con archivos de passwords en máquinas de varias compañías .*COM (USA)*, se pudo hallar un password de *"root"* en poco menos de 1 hr., de un total de 1594 passwords, de los cuales 50 se habían adivinado en los 15 primeros minutos y 90 de ellos después de 35 minutos [16].

3.2 Primeras Alternativas al Problema de Autenticación Basada en Passwords

En la sección anterior se ha mostrado, tomando como ejemplo el caso de *UNIX*, que el principal problema de la autenticación basada en passwords reside en la elección de passwords débiles por parte de los usuarios y que no se puede obligar a estos a elegir passwords criptográficamente fuertes. En términos más generales, en esta sección (inspirada en [11] (pp. 2)), se ilustra de manera sencilla la problemática de seguridad inherente a los protocolos de autenticación basada en passwords. Se inicia dando ejemplos de protocolos sencillos, sus debilidades y fortalezas. Después, se les agrega complejidad y propiedades adicionales hasta llegar a protocolos más viables y fuertes.

3.2.1 Ejemplo de un Protocolo Simple: Protocolo 1

1. A envía a B su identidad (A) y su password (P) en claro
2. B verifica P (comparándolo con su versión en claro o aplicando una función hash y comparando el resultado contra un archivo de resultados hash de passwords, almacenados en B)

Debilidad del Protocolo 1

1. Es vulnerable a ataques tipo réplica (Subsección 2.3.3 del Capítulo 2) en el cual el atacante obtiene P por monitoreo, y lo usa más tarde

para hacerse pasar por A ante B.

3.2.2 Ejemplo de un Protocolo Mejor: Protocolo 2

1. A envía a B su identidad (A), junto con un número aleatorio Ch_A (*quienvive*) que en este protocolo sirve como elemento de unicidad (Subsección 2.2.2 del Capítulo 2).
2. B envía su propio quienvive (Ch_B) a A
3. A hace algún cálculo basado en Ch_B y P. Envía ese cálculo a B, quien hace el mismo cálculo y verifica si coinciden los resultados

Fortaleza del Protocolo 2

1. Debido a que el quienvive es distinto para cada intento de autenticación, una respuesta capturada por un atacante es inútil para intentos posteriores, soportando ataques tipo réplica.

Debilidad del Protocolo 2

1. Un atacante I captura el resultado del cálculo, hecho en base a Ch y P, en una corrida del protocolo y hace un ataque por diccionario sobre P. Este ataque se ha utilizado para atacar exitosamente sistemas tipo $UNIX$
2. El protocolo 2 requiere que B conozca P y, por tanto, que B lo guarde en algún lugar, de algún modo
3. Aún cuando A y B transformen P por medio de una función hash antes de usarlo en el protocolo, el resultado tiene que guardarse en algún lugar en B. Un atacante I puede aplicar la misma transformación a passwords de prueba, lo que convierte el resultado hash almacenado en texto *equivalente a texto en claro* porque el atacante puede llegar a ese resultado almacenado a partir de conjeturar passwords en claro, por ejemplo mediante un ataque tipo diccionario. De esta manera puede romperse un sistema si el archivo de resultados hash se compromete o se filtra

3.2.3 Protocolo que supera deficiencias de los anteriores: Protocolo 3

La debilidad 3 del Protocolo 2 consiste en que el resultado que se guarda en B lugar de P puede ser comparado con resultados de aplicar la misma transformación a passwords de prueba, verificando de este modo si el password conjeturado es correcto o no. Es decir, el problema consiste en evitar que B almacene texto *equivalente a texto en claro*.

Una idea para superar esa debilidad sería que B guarde suficiente información para autenticar a A pero insuficiente información para suplantarlo. Una forma de lograr esto sería que A genere un secreto derivado de P y entonces aplique una función unidireccional a ese secreto para generar un *verificador*, el cual sería enviado a B. B entonces podría guardar ese verificador, sólo útil para construir un quienvive para A y para verificar una respuesta, pero que no pueda usarse en lugar de P para suplantar a A, y que no sea fácil intentar derivar P a partir del *conocimiento* de ese verificador. La idea es que un archivo de verificadores se pueda proteger tan fácil y efectivamente como un archivo de passwords, pero que el hecho de comprometer ese archivo de verificadores no sea tan catastrófico como compromter el archivo de passwords [11] (pp. 2).

En las siguientes dos secciones se revisará la evolución de las alternativas de este tipo, que muestran la existencia de alternativas y soluciones viables al difícil problema de lograr autenticación fuerte usando passwords débiles, hasta llegar al estado del arte actual en esta área.

3.3 Evolución: Alternativas Viables

Todo intento de solucionar un problema dado empieza por buscar alternativas que sin ofrecer una solución definitiva, presentan una mejora sustancial a lo que hasta ese momento se tiene y que perfilan un conjunto de ideas que posteriormente nutren y generan otras que llevan a soluciones reales. Este es el caso del problema de lograr autenticación fuerte usando passwords débiles. Esta sección describe los primeros intentos para resolver el problema.

3.3.1 Los Inicios

La investigación en protocolos de autenticación se vió fuertemente incrementada desde la publicación en 1978 del trabajo seminal de Needham y Schroeder [26], donde por primera vez se usa criptografía para autenticación y se introduce el concepto de núnico como número generado aleatoriamente para utilizarse una sola vez. En ese trabajo se propone una familia de protocolos de dos y tres partes para autenticación y distribución de llaves [25] (pp. 1).

Esos protocolos mostraron posteriormente contener algunas debilidades sutiles como el hecho de permitir usar una llave de sesión vieja en lugar de una nueva [27]. Esas debilidades fueron corregidas en la familia de protocolos presentados en [32], y esta familia de protocolos sirvió como base para el sistema Kerberos [4] desarrollado en el M.I.T.

Otro esfuerzo importante en la construcción de protocolos seguros fué el de Bird et al. [3, 34, 43] cuyo principal resultado es un protocolo de autenticación de dos partes, que muestra ser seguro contra ataques tipo combinación (Subsección 2.3.4 del Capítulo 2) [25] (pp. 1).

A este protocolo le siguió una familia de protocolos de tres partes para autenticación y distribución de llaves [34], que han sido implementados en un sistema de seguridad en red conocido como KryptoKnight [35] (pp. 1).

3.4 Protocolos de Intercambio de Llaves

La autenticación en el contexto de este trabajo involucra, por parte de los participantes en el protocolo, algún tipo de conocimiento o relación sobre un password. Ese password funciona algunas veces como llave secreta y puede usarse otras veces para derivar una llave de cifrado o de sesión que sirve para cifrar la comunicación entre las partes. Esa llave de sesión necesita ser acordada e intercambiada entre las partes involucradas. También en muchos casos, incluyendo este trabajo, cuando se habla de protocolos de autenticación se habla implícitamente de intercambio de llaves de sesión.

Por lo anterior, al hablar de la evolución de los protocolos de autenticación,

se tiene que mencionar la influencia y participación de los protocolos de intercambio de llaves en esa evolución, así como también los principales conceptos relacionados con estos protocolos.

Se conoce como *establecimiento de una llave* al proceso o protocolo por medio del cual una llave secreta compartida se hace disponible para dos o más partes, para uso criptográfico subsecuente. El proceso de establecer una llave puede subdividirse en *acuerdo* y *transporte* de la llave [50] (pp. 35 - 36). *Acordar* una llave involucra una técnica de establecimiento en la que un secreto compartido es derivado por dos o más partes en función de la información conque cada parte contribuye, o de la información asociada con cada una de esas partes. Idealmente ninguna de las partes puede predeterminar el valor resultante. *Transportar* una llave involucra una técnica de establecimiento donde una parte crea u obtiene un valor secreto y de manera segura lo transfiere a la otra u otras partes.

En este trabajo, protocolo de intercambio de llaves se usará como sinónimo de protocolo de establecimiento de llaves.

Los protocolos de intercambio de llaves que involucran autenticación, normalmente requieren una fase de inicialización (*"set-up"*) por medio de la cual se distribuyen llaves auténticas y posiblemente secretas. La mayoría de estos protocolos tienen como uno de sus objetivos la creación de llaves distintas para cada ejecución del protocolo. En algunos casos, la llave inicial predefine una llave fija que resultará cada vez que el protocolo es ejecutado. Los sistemas que involucran tales llaves estáticas son vulnerables a ataques por *llave conocida*.

Se dice que un protocolo es vulnerable a un *ataque por llave conocida* si el comprometer llaves de sesión pasadas permite a un atacante pasivo comprometer futuras llaves de sesión, o permite, en el futuro, la impostura por parte de un atacante activo [50] (pp. 496).

Este ataque efectuado sobre protocolos de intercambio de llaves es análogo a los ataques por *texto en claro conocido* sobre los algoritmos de cifrado. Una motivación para su consideración aquí es que en algunos ambientes (por ejemplo, debido a implementaciones y decisiones de ingeniería), la probabilidad de comprometer llaves de sesión puede ser mayor que la de comprometer llaves permanentes. Otra motivación es que cuando se usan técnicas criptográficas de moderada fortaleza, existe la posibilidad que con criptoanálisis

se puedan descubrir llaves de sesión pasadas.

En un protocolo de intercambio de llaves generalmente se desea que cada parte sea capaz de determinar la verdadera identidad de la otra parte que podría tener acceso a la llave. Esto implica prevenir que cualquier parte no autorizada pueda deducir la misma llave. En este caso se dice que la técnica proporciona *intercambio de llave seguro*. Esto requiere confidencialidad de la llave e identificación de las partes con acceso a ella. Este requerimiento de identificación difiere sutilmente, pero de manera importante, de la autenticación normal. En el caso del intercambio de llave seguro, se requiere conocer la identidad de las partes que pueden tener acceso a la llave, en lugar de la corroboración de que la comunicación actual ha sido establecida con tales partes [50] (pp. 491).

Se conoce como *autenticación de llave* la propiedad por medio de la cual una de las partes se asegura que ninguna otra parte distinta de la específicamente identificada, pueda tener acceso a una llave secreta particular. La autenticación de una llave es independiente de la posesión actual de tal llave por la otra parte legítima, o del conocimiento de tal posesión por la primera parte. No requiere involucrar acción alguna de la segunda parte involucrada. Debido a esta propiedad algunas veces la autenticación de llave se conoce más precisamente como *autenticación implícita de llave*. Se conoce como *confirmación de llave* a la propiedad por medio de la cual una parte se asegura que una segunda parte tiene la posesión actual de una llave secreta particular [50] (pp. 492).

Se llama *autenticación explícita de llave* la propiedad obtenida cuando se cumplen tanto la autenticación implícita de llave como la confirmación de llave. Un *protocolo de intercambio de llave autenticada* es un protocolo de intercambio que proporciona autenticación de llave [50] (pp. 492).

En un protocolo de intercambio de llaves que involucra autenticación, un requerimiento crítico es que el protocolo garantize que la parte cuya identidad se corrobora sea la misma parte con la que se intercambia la llave. Cuando esto no pasa, un atacante puede utilizar la ayuda de una parte autorizada para llevar a cabo la autenticación, y entonces suplantar a esa parte en el intercambio de llave (y comunicación subsecuente) [50] (pp. 493).

Los protocolos de intercambio de llaves obtienen como resultado secretos compartidos, los cuales se usan típicamente para derivar llaves de sesión.

Idealmente una llave de sesión es un secreto efímero. Es decir, su uso se restringe a un periodo de tiempo corto tal como una sesión única o una conexión para telecomunicación, después del cual se elimina todo vestigio de esa llave. Algunas motivaciones para usar llaves de sesión temporales son las siguientes [50] (pp. 494):

1. Limitar el texto cifrado (bajo una llave fija) disponible para ataques criptoanalíticos.
2. Limitar la divulgación con respecto a periodos de tiempo y cantidad de datos, en caso de compromiso de la llave de sesión
3. Evitar almacenamiento a largo plazo de gran número de llaves secretas (en el caso que una entidad se comunica con gran número de otras entidades), creando llaves sólo cuando realmente se requieren
4. Crear independencia entre comunicación para sesión y para aplicaciones
5. En la práctica, es deseable evitar el requerimiento de mantener información de estado de sesiones distintas

Al analizar protocolos de intercambio de llaves, debe considerarse el impacto potencial de comprometer información relacionada a las llaves, aún cuando tal compromiso no es normalmente esperado. En particular, deben considerarse los siguientes compromisos [50] (pp. 496):

1. Comprometer llaves permanentes (simétricas o asimétricas), si las hay
2. Comprometer llaves de sesión pasadas

Se dice que un protocolo tiene *seguridad perfecta hacia adelante* si el comprometer llaves permanentes no compromete llaves de sesión pasadas. La idea de seguridad hacia adelante (algunas veces llamada *protección de ruptura hacia atrás*) es que el tráfico previo sea asegurado de tal manera que no pueda ser utilizado en el tráfico actual. Esto puede ser proporcionado generando llaves de sesión por medio del algoritmo de establecimiento de llave exponencial de Diffie-Hellman [1], donde los intercambios exponenciales

[1] Sección A.11 del Apéndice A

se basan en llaves temporales. Si se comprometen llaves secretas permanentes, las sesiones futuras están, no obstante, sujetas a suplantación por un atacante activo [50] (pp. 496).

Los protocolos de intercambio de llaves han sido una plataforma para las recientes investigaciones en autenticación basada en passwords. La función de estos protocolos es establecer llaves frescas, por lo que sus objetivos de diseño se han centrado más en proporcionar *seguridad hacia adelante* que en proteger passwords débiles. Muchos de estos protocolos asumen que los secretos permanentes son criptográficamente fuertes y no requieren protección contra ataques tipo diccionario.

Los protocolos de intercambio de llave autenticada pueden dividirse en varias categorías. La distinción más importante es entre protocolos simétricos y asimétricos. Los primeros asumen que ambos lados poseen secretos permanentes idénticos y usan ese hecho como base para negociar nuevas llaves de sesión; los últimos usan un sistema basado en verificador para realizar la autenticación. Un sistema asimétrico es preferible para un sistema basado en passwords porque comprometer el verificador no compromete el password, mientras que en un sistema simétrico comprometer el password compromete de inmediato a las dos partes [11] (pp. 19).

Un buen mecanismo de autenticación basado en password buscaría combinar ventajas de los protocolos de intercambio de llaves (seguridad hacia adelante, intercambio de llave seguro) y de los protocolos basados en verificador (la información sobre el password no se revela al verificador), junto con otras características específicas de sistemas basados en passwords, como la protección de passwords débiles contra ataques tipo diccionario. Un enfoque extendido que junta un protocolo simple de intercambio de llaves con un protocolo de autenticación basado en verificador, fué usado por Goss [21] y luego por Bellovin y Merrit [7], y por Jablon [20] para obtener un sistema asimétrico de intercambio de llaves. La variante conocida como *MTI* [22] de Diffie-Hellman, usa el mismo enfoque de combinación de protocolos [11] (pp. 19).

Todos estos protocolos logran su seguridad ejecutando dos distintas corridas, una después de la otra, y combinando las dos llaves temporales de sesión, generadas una en cada ejecución, para formar la llave real. Es por esta razón que el intercambio asimétrico de llave autenticada tiende a imponer al menos dos veces la carga computacional de los intercambios de llave simétricos

convencionales [11] (pp. 19 - 20).

3.5 El Estado Actual

En la mayoría de los sistemas de seguridad actuales, la implementación de alternativas de autenticación basadas en passwords presenta serias debilidades de seguridad, que hacen que esas alternativas sean actualmente insatisfactorias y obsoletas. Esto se debe a que a la fecha existen soluciones viables al problema que no se han incorporado a los grandes sistemas abiertos. Estas soluciones son las que se conocen como métodos o protocolos fuertes. En esta sección se dará una breve visión tanto de las soluciones insatisfactorias en uso como de las soluciones fuertes que ya existen.

3.5.1 Alternativas Insatisfactorias en Uso

Hasta la fecha, muchas alternativas se han propuesto y usado; la mayoría aún se utilizan, aunque hayan mostrado ser inseguras [52]. Algunas de estas alternativas inseguras son las siguientes:

1. **Password en claro**

   Estos métodos predominan aún, tanto en *Internet* como en servicios como *telnet*, *ftp*, y cualquier otro donde el password se envía en claro en una sesión donde la información viaja sin cifrarse [11] (pp. 1). Estos métodos son vulnerables a atacantes que monitorean en la red [12, 42].

2. **Password mezclado**

   Estos métodos simplemente obscurecen el password, usando alguna transformación bien conocida o fácilmente descubrible. No son significativamente más fuertes que enviar el password en claro [52].

3. **Quienvive-Respuesta**

   Muchas variaciones de esta técnica se han usado desde principio de los 80's. El problema es que permiten ataques tipo diccionario cuando el password es débil. A pesar de esta limitación, son mejores que los

anteriores y se han usado para autenticación en *NetWare 3*, *Banyan VINES*, *LAN Manager*, y otros [49, 51].

4. **Kerberos**

   En **Kerberos V4 y V5** [4, 33], se cifra una ficha o boleto inicial usando el password como llave. Los datos de la ficha permiten ataques diccionario por texto verificable. Realmente, **Kerberos V4** es más débil que la opción quienvive-respuesta. Aún con la mejora de pre-autenticación en la **V5**, sigue siendo vulnerable a ataques diccionario [33].

5. **S/Key**

   S/Key (Sección A.6 del Apéndice A) [42] se refiere a un problema diferente en el sentido que puede usarse junto con programas originalmente diseñados para aceptar solamente passwords en claro. **S/Key** no evita ataques diccionario contra passwords débiles [52].

6. **Llave pública**

   En *NetWare 4* [49] y varios otros sistemas, el password se protege cifrándolo con una llave pública persistente, es decir con una llave que permanece almacenada en algún lado. No obstante, el confiar en una llave pública persistente puede causar problemas. Cuando la llave pública se envía en claro, como en *NetWare 4*, el engaño o suplantación es posible. Cuando la llave pública se distribuye previamente, como en *SPX* [53], o se pre-certifica, como en una sesión anónima *SSL* (Sección A.7 del Apéndice A), hay problemas complejos adicionales impuestos por la necesidad de certificación y revocación de la llave [52].

7. **Firmas Digitales**

   El uso de firmas digitales (Sección A.5 del Apéndice A) en los navegadores ha creado un nuevo estilo de autenticación que muchos consideran la solución final. La seguridad de las firmas digitales depende de la seguridad de la criptografía de llave pública, la que a su vez depende de la seguridad de la llave privada del usuario. Cuando en un navegador se requiere un certificado digital para autenticar una firma, automáticamente se genera una llave privada para el usuario y esta llave se guarda en el disco de su computadora. Durante este proceso de generación de llaves, al usuario se le pide un password para cifrar la llave privada antes de guardarla en disco [54]. Desafortunadamente este esquema no es seguro debido a que el password proporcionado por

el usuario es débil y puede ser atacado por diccionario para conocer la llave privada. Por otro lado, la llave privada es tan segura como el software que la manipula, y existen numerosas debilidades de seguridad en el software de los navegadores que pueden ser explotadas para recuperar la llave de la memoria de la máquina después de que esta ha sido descifrada [54].

Todos los métodos anteriores son insatisfactorios en el sentido de que ahora existen soluciones más fuertes. De todas las alternativas anteriores, sólo los métodos de llave pública intentan evitar ataques diccionario fuera de línea. No obstante, dado que se basan en llaves públicas persistentes, son vulnerables a ataques por suplantación de identidad sobre el propietario de la llave [52].

Una alternativa es utilizar soluciones fuertes que no dependan de que el usuario guarde llaves permanentes. Los esquemas de secreto memorizado presentan ventajas sobre los esquemas de secreto almacenado por ser menos vulnerable a robo o reemplazo malicioso. Un método fuerte agrega un factor independiente de fuerza a la base de la autenticación. Todos los métodos fuertes utilizan técnicas criptográficas de llave pública, pero en lugar de llaves almacenadas usan llaves temporales [52]. Las parejas de llaves temporales, pública y privada, son creadas aleatoriamente, utilizadas una sola vez, y destruídas tan pronto como la autenticación termina. El primer método fuerte utilizó el password como llave simétrica con un cuidadoso diseño del protocolo para evitar ataques tipo diccionario.

3.5.2 Soluciones Fuertes

En 1989, Gong et al. presentaron un trabajo [6] que abrió las puertas a un gran número de problemas asociados con protocolos basados en passwords en sistemas de autenticación. Este trabajo introdujo el tipo de ataque conocido como texto verificable (Subsección 2.3.4 del Capítulo 2), del cual los ataques por texto en claro conocido forman un subconjunto. A pesar de que los ataques de este último tipo son relativamente fáciles de detectar y corregir, los ataques por texto verificable son, en general, más difíciles de evitar y mucho más sutiles [36]. En ese trabajo los autores presentan algunos ejemplos de protocolos criptográficos resistentes a ataques por texto verificable y sugieren una metodología de prueba de protocolos para detectar

vulnerabilidades a este tipo de ataques. Posteriormente, los mismos autores, en otro trabajo [1] refinan estas mismas ideas, generalizan el concepto de texto verificable y describen técnicas para evitar este tipo de ataques.

En 1992 Bellovin y Merritt [17] presentaron un nuevo protocolo conocido como *EKE*(*"Encrypted Key Exchange"*), que usando una combinación de criptografía convencional y de llave pública, logra autenticación, resistiendo ataques tipo diccionario al no proporcionar a los atacantes suficiente información para poder verificar un password. También realiza intercambio de llaves, de tal modo que ambas partes pueden cifrar sus transmisiones una vez que la autenticación se ha realizado.

La idea básica atrás del protocolo *EKE* es que si se cifra una cadena de bits completamente aleatoria usando el password como llave, no hay manera de que un atacante pueda verificar, probando con diferentes passwords, si ha descubierto o no una llave de sesión puesto que el resultado de descifrar la cadena aleatoria de bits no tendrá ningún sentido para el atacante.

El punto es entonces garantizar que la prueba de passwords no pueda ser verificada a partir de la información que se intercambia durante la ejecución del protocolo. Esto se logra en los protocolos *EKE* cifrando una llave pública generada aleatoriamente, de tal modo que todo intento de romperla, probando passwords, dé como resultdo un número aleatorio inútil de verificar por carecer de significado para el atacante. La llave de sesión que se intercambia puede, de este modo, cifrarse usando como llave el password compartido. Romper la llave pública se considera computacionalmente infactible.

Desde entonces, a partir de *EKE*, se ha desarrollado una familia de protocolos, muchos de ellos más fuertes que el original y con nuevas propiedades adicionales. Por ejemplo, *EKE-DH* [18] y *SPEKE* [19] agregan lo que se conoce como seguridad hacia adelante (Sección 3.4 del Capítulo 3). En [1] se describe un conjunto de protocolos llamados de *llave pública secreta* que consiguen los mismos objetivos que *EKE*.

A la fecha, la familia de protocolos *EKE* representa el nivel más fuerte de protocolos de autenticación disponible [11] (pp. 3). La mayor falla de ellos consiste en que aún sufren de *equivalencia a texto en claro* (Subsección 3.2.2 del Capítulo 3), requiriendo que cliente y servidor tengan acceso al mismo password o resultado hash de él [11] (pp. 3).

Existe una variante de EKE conocido como *"Augmented EKE"* o $AEKE$ [7] el cual hace de EKE un protocolo basado en un verificador distinto del password, por ejemplo una función unidireccional de él. Esta modificación, no obstante, destruye la seguridad hacia adelante [18].

Los protocolos de la famila $AEKE$ emplean el intercambio de secretos, en lugar del tradicional secreto compartido y no usan ninguna forma de cifrado simétrico para lograr su seguridad [18].

Trabajos recientes han extendido la familia EKE hacia el problema de posesión de información equivalente a texto en claro en los archivos de passwords [20]. $B\text{-}SPEKE$ es un ejemplo de tal método. Estos protocolos agregan otra ronda de intercambio de llave con el fin de verificar la posesión, por parte del cliente, del password actual; esto, como oposición al posible robo de un verificador almacenado en un archivo de passwords. Esto corrige un problema mayor de EKE a cambio de aumentar sustancialmente el tiempo de ejecución y la complejidad computacional del protocolo resultante [11] (pp. 3).

Aunque esta área de investigación es relativamente joven, es muy dinámica, y constantemente están apareciendo trabajos con contribuciones importantes. Una excelente manera de mantenerse al tanto de los avances y cambios en esta área son las referencias listadas en la Sección B.1 del Apéndice B de este trabajo.

Dentro de las más recientes contribuciones al área está el trabajo [58] de M. Roe, B. Christianson, y D. Wheeler donde presentan tres protocolos basados en los sistemas de llave pública RSA (*"Rivest, Shamir, Adleman"*) [28], Diffie-Hellman, y ElGamal respectivamente. Estos protocolos pretenden ser más simples y rápidos que sus predecesores, garantizando una seguridad ligeramente superior; en particular, no cifran la llave de sesión k, y por tanto no imponen ninguna restricción al uso de k en otros protocolos. A estos protocolos basados en sistemas de llave pública, los autores los llaman genéricamente $S3P$ (*"Strong Secret Sharing Password Protocol"*).

También recientemente, S. Halevi y H. Krawczyk publicaron un trabajo [61] donde estudian protocolos de este tipo en escenarios asimétricos en los que el servidor posee una pareja de llaves pública/privada, y el cliente tiene sólo un password P como llave de autenticación. El análisis muestra a esos protocolos como óptimamente resistentes a ataques diccionario fuera de línea.

Además, para autenticación de usuarios, presentan protocolos que proporcionan autenticación de dos partes, intercambio de llave autenticada, defensa contra compromisos del servidor, y anonimato del usuario. También realizan una prueba que muestra que las técnicas de llave pública son inevitables para protocolos basados en passwords que pretendan resistir ataques diccionario fuera de línea. Introducen la noción de *passwords públicos* que usan en tales protocolos para situaciones donde la máquina del cliente no cuenta con los recursos para validar la llave pública del servidor. Estos passwords públicos se pueden conceptualizar como compendios de la llave pública del servidor; el usuario no necesita memorizarlos, puede escribirlos o almacenarlos en algún lado, y puede usarlos sin necesidad de dispositivos especiales de cómputo.

Actualmente existe otra corriente de investigación de protocolos basados en passwords que ya no utilizan criptografía para intercambiar información de autenticación e intercambio de llaves de sesión. En lugar de eso, se usan relaciones matemáticas predefinidas para combinar valores temporales intercambiados con parámetros que se establecen como passwords, pero que no son el password compartido P. El password P ya no se intercambia de ningún modo. Estos protocolos describen un enfoque conocido como "intercambio de secreto" en el que cada parte calcula un secreto y entonces aplica una función unidireccional a ese secreto para generar un verificador. Ese verificador es enviado a la otra parte. Aún cuando es importante proteger el verificador para prevenir ataques tipo diccionario, el robo del verificador no sería suficiente para suplantar al usuario; se requeriría aún el password correspondiente.

Resultados concretos y recientes en esta dirección han sido los siguientes. En Abril de 1997, Stefan Lucks de la Universidad de Gottingen, publicó un nuevo protocolo llamado *OKE* (*"Open Key Exchange"*) [41], el cual ya no utiliza criptografía para intercambiar llaves públicas, las cuales, en este protocolo, viajan en claro.

En Julio de 1997, Tom Wu de la Universidad de Stanford, publicó varios protocolos conocidos colectivamente como *SRP* (*"Secure Remote Password Protocols"*) [11] los cuales resisten ataques tipo diccionario, ofrecen seguridad hacia adelante y almacenan los passwords de forma que no contienen información equivalente a texto en claro al password mismo. Estos protocolos combinan técnicas basadas en pruebas de conocimiento cero [2] [46] con

[2] Sección A.8 del Apéndice A

protocolos asimétricos de intercambio de llaves, mejorando el rendimiento que ofrecen *EKE* o *B-SPEKE*.

Debido a que el enfoque de nuestro trabajo es hacia protocolos de autenticación e intercambio de llaves que utilizan alguna técnica criptográfica para lograr sus objetivos, la corriente antes mencionada no será tratada con detalle, como lo haremos con los protocolos criptográficos.

El pobre rendimiento que logran los protocolos llamados fuertes, ha sido con frecuencia un obstáculo para adoptarlos como base en implementaciones concretas; los protocolos *AEKE* [7] y los protocolos extendidos *SPEKE* y otros, descritos en [20] son demasiado lentos para necesidades de autenticación frecuentes y rápidas. Una mejora de 3 a 1.5 segundos en el rendimiento puede hacer la diferencia entre una solución intolerable y una aceptable [11] (pp. 3).

Capítulo 4

Protocolos Fuertes

En este capítulo se describen y analizan los principales protocolos fuertes que han surgido hasta la fecha como soluciones viables al problema de autenticación basada en passwords. Se mencionan las principales ideas que dieron origen a cada uno de estos protocolos o familias de ellos, las técnicas que utilizan, las debilidades y fortalezas que muestran tener, los ataques que resisten, los posibles ataques a los que son vulnerables, y las mejoras que son susceptibles de aceptar, entre otros puntos de análisis.

Se inicia este capítulo presentando la familia de protocolos conocida como *LGSN*(*Lomas, Gong, Saltzer, Needham*) [1, 6], las iniciales de sus autores, por ser considerados como los primeros protocolos fuertes, surgidos entre 1989 y 1992. Después, se analiza la familia de protocolos genéricamente conocida como *EKE* (*"Encrypted Key Exchange"*) [7, 17] y sus derivaciones. Finalmente, se presentan los métodos conocidos como extendidos por ser derivaciones y extensiones, principalmente de la familia *EKE*.

4.1 Protocolos LGSN (Lomas, Gong, Saltzer, Needham)

Los autores [6] de estos protocolos buscan diseñar protocolos de autenticación asumiendo que el hecho de elegir passwords débiles por parte de los

usuarios es algo que no se puede evitar, y tratan de que ese hecho no sea importante para la seguridad de esos protocolos.

Una inspiración para este enfoque es observar que los cajeros automáticos generalmente usan 4 dígitos como passwords y no parecen particularmente vulnerables a ataques por diccionario. La razón es que aunque se pueden hacer alrededor de 9999 intentos, el sistema confisca la tarjeta al usuario después del tercer intento fallido [8]. La clave de la no atacabilidad del pequeño espacio de passwords del cajero, es que la prueba no puede hacerse *de manera aislada; con cada intento se invoca una parte del sistema que* registra los intentos incorrectos y genera una alarma si el número de intentos excede el límite permitido.

Otra inspiración consiste en observar que si se cifra una cadena completamente aleatoria de bits, un atacante no tiene ningún modo de verificar si ha descubierto una llave, porque al descifrar esa cadena no encuentra ningún significado ya que obtiene como resultado un conjunto aleatorio de bits. Así, un buen criterio para diseñar protocolos de autenticación consiste en usar passwords, o llaves derivadas de ellos, únicamente para cifrar datos impredecibles para el atacante.

En las siguientes subsecciones se hace un análisis de los principales protocolos de esta familia. El análisis de estos protocolos se basa fundamentalmente en [1, 6, 36, 55], y para ello se usa el modelo y la notación especificados en la Sección 2.2 del Capítulo 2 de este trabajo.

4.1.1 LGSN Original

El protocolo *LGSN* original [6] (pp. 4) tiene como objetivo específico, lograr autenticación mutua de dos partes resistiendo ataques tipo texto verificable (Subsección 2.3.4 del Capítulo 2).

En particular, este protocolo asume la existencia de un servidor de llaves S con el cual los clientes A y B comparten, por separado, una llave. Si las partes A y B son usuarios y no algún tipo de dispositivo tal como una computadora, esta llave puede ser una derivación algorítmica del password P del usuario, o el password mismo. Es por esta razón por la que se usa la notación P_A o P_B, según el caso. El protocolo usa criptografía de llave

pública y de llave secreta. S mantiene su llave privada en secreto. A y B no comparten ningún secreto entre sí. Los pasos del protocolo, son los siguientes:

1. A selecciona los núnicos N_{A_1} y N_{A_2}, el confundidor C_A, y la instántanea T_A la cual se cifra usando como llave P_A que comparte con S. Todos estos elementos se cifran junto con la identidad de A y la de B, usando como llave de cifrado la llave pública de S, K_S^p. A envía el resultado a S

   $A \longrightarrow S : E_{K_S^p}(\text{A, B, } N_{A_1}, N_{A_2}, C_A, E_{P_A}(T_A))$

2. Al recibir el paquete cifrado anterior, S utiliza su llave privada para descifrarlo; utiliza P_A para descifrar la instantánea T_A. Verifica que T_A esté dentro del rango de tiempo correcto. S envía a B la identidad de A y de B

   $S \longrightarrow B : A, B$

3. Al recibir el mensaje anterior, B hace lo mismo que hizo A en el paso 1, creando su paquete cifrado y enviándolo a S

   $B \longrightarrow S : E_{K_S^p}(\text{B, A, } N_{B_1}, N_{B_2}, C_B, E_{P_B}(T_B))$

   Hasta aquí, se usa criptografía de llave pública. Los siguientes intercambios se hacen utilizando criptografía de llave secreta; primero usando como llaves las compartidas con S y luego la llave de sesión k, que S selecciona y distribuye.

4. S selecciona la llave de sesión k, calcula la operación XOR sobre k y N_{A_2}; al resultado le concatena N_{A_1}. Cifra todo esto usando P_A como llave. Envía este paquete cifrado a A

   $S \longrightarrow A : E_{P_A}(N_{A_1}, k \bigoplus N_{A_2})$

5. S hace lo mismo que en el paso anterior, pero ahora con B

   $S \longrightarrow B : E_{P_B}(N_{B_1}, k \bigoplus N_{B_2})$

   Al recibir el mensaje 4, A utiliza P_A compartido con S para descifrarlo. Verifica que N_{A_1} y N_{A_2} sean correctos; es decir, que correspondan a los que A envió en el mensaje 1. Si todo es correcto, acepta k como llave de sesión para los siguientes intercambios. Lo mismo hace B al recibir el mensaje 5.

De aquí en adelante los intercambios tienen como objetivo realizar el saludo (Subsección 2.2.1 del Capítulo 2) entre A y B. Para ello, intercambian quienvives y resultados hash de ellos, calculados usando funciones hash previamente acordadas, f_1 y f_2, y cifrándolos con la llave de sesión k.

6. $B \longrightarrow A : E_k(Ch_A)$
7. $A \longrightarrow B : E_k(f_1(Ch_A),\ Ch_B)$
8. $B \longrightarrow A : E_k(f_2(Ch_B))$

Análisis

El análisis de este protocolo se hace en términos de sus fortalezas, ataques que puede sufrir, las formas de defensa contra esos ataques y las debilidades inherentes al protocolo. Este análisis está inspirado en [1, 6, 36, 55].

Fortalezas

1. La llave pública de S, K_S^p, sólo se usa para cifrar el requerimiento inicial a S, de tal modo que un sistema sólo necesita una de tales llaves
2. S descifra los mensajes 1 y 3 usando su llave privada. Verifica la identidad de A y B descifrando $(T_A)_{P_A}$ y $(T_B)_{P_B}$. Si el descifrado no produce el tiempo actual (dentro del ajuste permitido del reloj), S registra una falla; si produce el tiempo correcto, responde con los mensajes 4 y 5
3. Los mensajes 4 y 5 contienen N_{A_1} y N_{B_1}, respectivamente, como prueba de que los mensajes 1 y 3 fueron descifrados correctamente. Si S no puede descifrar el mensaje 1, entonces el mensaje 4 no es enviado, de tal modo que P_A queda asegurado contra posibles ataques
4. El mensaje 4 y 5 contienen N_{A_2} y N_{B_2}, respectivamente, por dos razones: para proteger a A y B contra un ataque sobre P_A y P_B por parte de B y A quienes ya conocen el valor de k; y, la más importante, evitar que los mensajes 6 y 7 o 7 y 8 sean explotados para verificar k por algún atacante, lo que le permitiría verificar P_A o P_B

5. Los mensajes 6 y 7 contienen un quienvive Ch_A y la respuesta $f_1(Ch_A)$ respectivamente, lo cual permite que la llave k, si se conjetura, sea verificada. Similarmente, los mensajes 7 y 8, cuando se toman en pareja, contienen texto en claro verificable cifrado con k. Lo que hace infactible este ataque es el hecho de que k se origina en el servidor, y por lo tanto se supone bien seleccionada

6. El protocolo incluye una instantánea. Si el atacante no conoce el tiempo exactamente, esto incrementa el número de pruebas necesarias para verificar el éxito de una conjetura sólo en un factor pequeño. No obstante, si la instantánea se lleva a una precisión mucho más grande de lo que el atacante puede conocer, entonces los bits de bajo orden de la instantánea también pueden actuar como confundidores de un ataque por conjetura

Ataques

Ataque por Participante Legítimo

Aunque se puede asumir que las partes legítimas involucradas en el protocolo confían una en la otra, las llaves que cada participante comparte con el servidor son distintos, y por tanto no es adecuado extender esa confianza al grado de compartir esas llaves, o confiar que B no intentará adivinar la llave de A utilizando los mensajes de una ejecución exitosa anterior.

Una protección contra este ataque también aseguraría que B no compromete el password de A, al comprometer el suyo.

Los confundidores C_A y C_B sirven para defender a A de B y viceversa. Alguno, de B o A, que intente adivinar P_A o P_B, puede intentar construir los mensajes 1 o 3 sin descifrarlos. Para ver esto, considérense los siguientes mensajes: $1'$ que reemplaza al mensaje 1, omitiendo C_A; y el mensaje 4 que permanece sin cambios

$1'.\ A \longrightarrow S : E_{K_S^p}(\mathrm{A},\ \mathrm{B},\ N_{A_1},\ N_{A_2},\ E_{P_A}(T_A))$

$4.\ S \longrightarrow A : E_{P_A}(N_{A_1},\ k \oplus N_{A_2})$

Suponiendo que el atacante es B, este conoce k y ha registrado los mensajes $1'$ y 4. Con esta información en su poder, conjetura P_A e intenta descifrar

el mensaje 4 usando esta conjetura, probando con distintos N_{A_1} y N_{A_2}. Una forma de verificar la conjetura es comparando sus resultados con el mensaje 4 registrado. Otra forma es calculando el correspondiente texto cifrado de $1'$ y comparándola con la copia interceptada del mensaje $1'$; si las dos versiones de texto cifrado son idénticas, la conjetura de B sobre P_A ha sido verificada. Una manera de evitar esta última verificación es usar el confundidor C_A, número aleatorio que no se ha usado antes, el cual confunde este tipo de ataque que supone que todos los bits del mensaje cifrado 1 dependen de todos los bits del texto en claro de ese mensaje. El confundidor C_B, similarmente protege a P_B de conjeturas hechas por A.

Aunque el ataque anterior es poco probable por la cantidad de cómputo requerido al tener que probar distintos N_{A_1} y N_{A_2} para una sola conjetura de P_A o P_B, ilustra la función protectora de los confundidores.

Ataque por Texto Verificable

En 1993, Tsudik y Van Herreweghen [36] mostraron que el Protocolo *LGSN* puede ser atacado por texto verificable (Subsección 2.3.4 del Capítulo 2), ya sea por un participante en el protocolo (A o B), o por atacantes externos. Este ataque se basa en un hecho común en el diseño de servidores: dado que no existe ningún método perfecto de sincronización de relojes, se puede asumir un máximo permisible para el ajuste del reloj, Δ. También se puede suponer que el servidor no registra ni guarda instantáneas generadas por cada participante en el protocolo.

El ataque supone que el adversario registra una corrida completa del protocolo entre A, B y S. Entonces, procede a replicar el mensaje 1 dentro del intervalo de tiempo permitido, es decir, entre T_A y $(T_A + \Delta)$. S, no reconociendo la réplica, contesta a A (en el mensaje 4) con:

$4'.\ S \longrightarrow A : E_{P_A}(N_{A_1}, k' \oplus N_{A_2})$

El atacante previamente tiene registrado:

$4.\ S \longrightarrow A : E_{P_A}(N_{A_1}, k \oplus N_{A_2})$

Si el atacante es un participante en el protocolo (es decir, B), aquí se detiene. Conociendo k y k', B puede ahora iterar a través del espacio de llaves de

P_A y, para cada conjetura, descifrar ambos mensajes; realizar la operación $\bigoplus$ en la segunda parte del mensaje y comparar el resultado con $(k \bigoplus k')$[1]. Una coincidencia indica la conjetura probablemente correcta de P_A.

Si el atacante es externo se requiere hacer un poco más. Además de replicar el mensaje 1 registrado con anterioridad, el atacante también intercepta la comunicación entre S y B en el mensaje 2 y, pretendiendo ser B, replica con un mensaje 3 previamente registrado:

3. $B \longrightarrow S : E_{K_S^p}(\mathrm{B}, \mathrm{A}, N_{B_1}, N_{B_2}, C_B, E_{P_A}(T_B))$

Una vez más, ya que T_B es aún válido, S contesta a B con el mensaje 5′:

5′. $S \longrightarrow B : E_{P_B}(N_{B_1}, k' \bigoplus N_{B_2})$

Este mensaje también es interceptado por el adversario, quien previamente tiene registrado:

5. $S \longrightarrow B : E_{P_B}(N_{B_1}, k \bigoplus N_{B_2})$

En este punto, el atacante tiene que iterar a través de todas las posibles elecciones de P_A y P_B. Para cada par de conjeturas (P'_A, P'_B), el atacante descifra los mensajes 4, 4′, 5 y 5′. Luego, realiza la operación $\bigoplus$ con las segundas mitades de 4 con 4′ y de 5 con 5′ para obtener una pareja de valores *candidatos* de $(k \bigoplus k')$. Si los dos valores coinciden, esto indicará la conjetura probablemente correcta de ambas llaves. Este ataque es posible porque aún cuando el atacante no conoce k, k' o $(k \bigoplus k')$, puede explotar lo particular de la construcción del mensaje y extraer suficiente redundancia para que el ataque tenga éxito.

Defensa

Los ataques mostrados se vuelven infactibles si S implementa un mecanismo de detección de réplicas. En [1], Gong et al. establecen explícitamente este requerimiento, sugiriendo que S guarde los mensajes 1 y 3 durante un tiempo igual al máximo ajuste del reloj. La motivación de esta sugerencia es evitar el ataque descrito antes.

[1] Ya que $(N_{A_2} \bigoplus k' \bigoplus N_{A_2} \bigoplus k) = k' \bigoplus k$

Debilidades

1. Entre las debilidades importantes del protocolo *LGSN* están las consecuencias negativas del uso de instantáneas para autenticación.

   Un peligro notable del uso de instantáneas es discutido por Gong en [37], el cual involucra una falla en la inicialización del reloj de la parte generadora de la instantánea (que puede ser causada accidental o maliciosamente), de tal manera que el reloj esté incorrectamente definido para un tiempo futuro $T_f = (T_c + \Delta)$, donde T_f es un tiempo futuro dado, T_c es la lectura actual del reloj, y Δ es el máximo permitido para el ajuste del reloj. Un mensaje de autenticación relacionado a una instantánea que corresponda a esta lectura del reloj, será rechazado por la otra parte y el reloj puede eventualmente ser reinicializado a su operación normal.

   No obstante, el mensaje erróneo puede ser registrado por un atacante, quien aguardará pacientemente hasta que el mensaje registrado "madure"; es decir, hasta que el tiempo actual (la lectura del reloj) se haga $T_c = T_f$, de tal manera que el tiempo que registra la instántanea en el mensaje de la ejecución actual coincida con el tiempo registrado en la instantánea del mensaje capturado con anterioridad por el atacante. Entonces, el atacante simplemente replica el mensaje registrado y obtiene la autenticación deseada. No hay certeza [36] de que este tipo de ataque pueda ser detectado por la mayoría de esquemas de detección de réplicas, incluyendo el que se sugiere en [1].

2. Otra debilidad consiste en el costo de mantener un mecanismo de detección de réplicas en el servidor. La principal razón de esta necesidad, sugerida en [1], es la protección contra ataques por texto verificable mencionado antes. Esta protección involucra medidas fuera del protocolo, es decir, el estado en el servidor no está representado en el protocolo.

   Sin un mecanismo de detección de réplicas, el protocolo *LGSN* parece difícil de corregir. Se puede intentar aplicar una corrección rápida modificando el mensaje 4 de la siguiente manera:

   4. $S \longrightarrow A : E_{P_A}(N_{A_1}, E_{N_{A_2}}(k))$

   No obstante, los autores [6] suponen que los núnicos generados por el usuario no pueden usarse como llaves de cifrado. Las razones para esta supocisión no son totalmente claras, sobre todo tomando en cuenta que

los usuarios (sus computadoras) pueden seleccionar núnicos razonablemente aceptables en términos de aleatoriedad y fortaleza criptográfica.

3. Otra debilidad del protocolo *LGSN* es que requiere demasiadas operaciones de cifrado de llave pública. Requiere cifrar 6 campos con la llave pública de S. (Un total de 12 campos si se toma en cuenta la distribución de la llave de B). El costo del cifrado con llave pública no es despreciable.

   Para superar esta debilidad, muchos criptosistemas cifran datos en bloques relativamente grandes (por ejemplo, un tamaño de bloque razonable para el *RSA* [28] es entre 512 y 700 bits [36] (pp. 4)). Por tanto, la longitud del mensaje no importa, siempre que esté dentro del tamaño de bloque usado por el criptosistema.

4. Una debilidad más es que la distribución de llaves está combinada con la autenticación. Los mensajes 4 y 5 incluyen alguna redundancia en la forma de N_{A_1} y N_{B_1} respectivamente, lo que asegura a A y B la integridad y frescura del mensaje. Esta certeza no es estrictamente requerida, ya que la integridad y frescura de los mensajes 4 y 5 está implícitamente probada por el éxito de la siguiente autenticación entre A y B usando k.

5. Otra debilidad es que A y B tienen que seleccionar 3 números aleatorios cada uno. Generar estos números en general no es costoso. Sin embargo, muchos generadores de números aleatorios se basan en cifrado [38] (pp. 369 - 428) y cada número aleatorio puede costar una o más operaciones de cifrado. En este caso, es deseable minimizar el número de núnicos y confundidores que se necesitan en cada ejecución del protocolo.

6. El usar criptosistemas de llave pública no elimina la necesidad de mantener un secreto. En este protocolo el secreto de S es una llave privada; en el caso de A y B, el secreto es un password.

4.1.2 LGSN Reducido

Los diseñadores presentaron el protocolo *LGNS* Original [6] (pp. 4) para propósitos de demostración, motivo por el cual no está optimizado en términos de número de mensajes. En un nuevo trabajo [1] (pp. 10), presentan una versión que minimiza el número de mensajes.

Los mensajes intercambiados entre las partes fueron explicados con detalle en el protocolo *LGNS* Original por lo que sólo se harán las observaciones pertinentes cuando sea necesario, ya que básicamente los mensajes son los mismos. El protocolo *LGSN* Reducido es el siguiente:

1. A diferencia del protocolo *LGSN* original, en esta versión A envía el quienvive Ch_A a B en el primer mensaje, aún cuando la réplica de este mensaje se espera hasta el paso 4

   $A \longrightarrow B : E_{K_S^p}(\text{A, B, } N_{A_1}, N_{A_2}, C_A, E_{P_A}(T_A)), Ch_A$

2. Este paso integra los pasos 2 y 3 del protocolo *LGSN* Original

   $B \longrightarrow S : E_{K_S^p}(\text{A,B,}N_{A_1},N_{A_2},C_A,E_{P_A}(T_A)), E_{K_S^p}(\text{B,A,}N_{B_1},N_{B_2},C_B,E_{P_B}(T_B))$

3. En este paso se integran los pasos 4 y 5 del *LGSN* Original

   $S \longrightarrow B : E_{P_A}(N_{A_1}, k \bigoplus N_{A_2}), E_{P_B}(N_{B_1}, k \bigoplus N_{B_2})$

4. A este paso se integra el paso 7 del protocolo *LGSN* Original

   $B \longrightarrow A : E_{P_A}(N_{A_1}, k \bigoplus N_{A_2}), E_k(f_1(Ch_A), Ch_B)$

5. Este paso es el mismo que el paso 8 del *LGSN* Original, único que no cambia

   $A \longrightarrow B : E_k(f_2(Ch_B))$

Análisis

El análisis de este protocolo es esencialmente el mismo que para el protocolo *LGSN* Original. El quienvive Ch_A enviado de A a B en el primer mensaje, viaja en claro dentro del bloque porque el hecho de conocer ese texto en claro no ayuda a conjeturar P_A o P_B.

4.1.3 LGSN con Núnicos

No en todos los sistemas se puede asumir la existencia de relojes sincronizados. Para superar esta carencia, los autores presentan una versión conocida como *LGSN* con Núnicos [1] (pp. 11), debido a que usa también un núnico N_S seleccionado por el servidor, y utiliza la técnica conocida como quienvive-respuesta (Sección 1.4 del Capítulo 1) para establecer la puntualidad de los mensajes. Esta versión toma como base el protocolo *LGSN* Reducido descrito anteriormente. El protocolo es el siguiente:

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: A, B, N_S
3. $A \longrightarrow B$: $E_{K_S^v}$(A, B, N_{A_1}, N_{A_2}, C_A, $E_{P_A}(N_S)$), N_S, Ch_A
4. $B \longrightarrow S$: $E_{K_S^v}$(A,B,N_{A_1},N_{A_2},C_A,$E_{P_A}(N_S)$), $E_{K_S^v}$(B,A,N_{B_1},N_{B_2},C_B,E_{P_B}(N$_S$
5. $S \longrightarrow B$: $E_{P_A}(N_{A_1}, k \oplus N_{A_2})$, $E_{P_B}(N_{B_1}, k \oplus N_{B_2})$
6. $B \longrightarrow A$: $E_{P_A}(N_{A_1}, k \oplus N_{A_2})$, $E_k(f_1(Ch_A), Ch_B)$
7. $A \longrightarrow B$: $E_k(f_2(Ch_B))$

Análisis

El análisis de seguridad del protocolo no cambia con respecto al original. Esta versión agrega una ronda extra de intercambio entre A y S, en la que S selecciona y utiliza un núnico N_S que también sirve para validar la ejecución actual del protocolo. Con esto, S puede ahora detectar réplicas de mensajes en los que ahora se usa N_S en lugar de T_A y T_B. Nótese que A envía N_S a B en el mensaje 3.

Una debilidad de esta versión está en el hecho de que necesita dos mensajes adicionales y requiere que S mantenga información de estado. T_A se reemplaza por el núnico de S, el cual también debe mantener el estado de su núnico (N_S) durante el tiempo transcurrido entre su uso y la réplica de A. Si varios requerimientos llegan uno detrás de otro, S podría tener que

guardar el estado de muchos núnicos en su memoria principal lo que representa un requerimiento extra de memoria. Esto además, involucra el riesgo de pérdida de información de estado si ocurre una falla en el servidor.

4.1.4 LGSN con Distribución de Llaves Públicas

Algunas veces los usuarios pueden olvidar la llave pública de S, aún cuando la escriban en alguna parte (se puede perder) o la guarden en una calculadora (se puede quedar sin pilas). Una idea para evitar este problema es tener una versión del protocolo *LGSN* que integre una distribución de llaves para A y B que evite la distribución previa de la llave pública de S y el problema derivado de la posibilidad de que A o B pudieran olvidar o perder dicha llave.

Las llaves que S distribuye en este protocolo se puede asumir que son llaves públicas que S genera por parejas de pública-privada para A y B. S mantiene las correspondientes llaves privadas de A y B; además, sigue compartiendo con A y B el conocimiento sobre las llaves P_A y P_B respectivamente.

Este protocolo se basa en la versión reducida de *LGSN* y también se conoce como *LGSN con Llaves Públicas Secretas* [1] (pp. 11 - 12) debido a la novedosa forma secreta de utilizar llaves públicas.

El protocolo es el siguiente:

1. $A \longrightarrow S$: A, B
2. $S \longrightarrow A$: A, B, N_S, $E_{P_A}(K_A^p)$, $E_{P_B}(K_B^p)$
3. $A \longrightarrow B$: $E_{K_A^p}$(A, B, N_{A_1}, N_{A_2}, C_A, $E_{P_A}(N_S)$), N_S, Ch_A, $E_{P_B}(K_B^p)$
4. $B \longrightarrow S$: $E_{K_A^p}$(A,B,N_{A_1},N_{A_2},C_A,$E_{P_A}(N_S)$), $E_{K_B^p}$(B,A,N_{B_1},N_{B_2},C_B,$E_{P_B}(N_S)$)
5. $S \longrightarrow B$: $E_{P_A}(N_{A_1}, k \oplus N_{A_2})$, $E_{P_B}(N_{B_1}, k \oplus N_{B_2})$
6. $B \longrightarrow A$: $E_{P_A}(N_{A_1}, k \oplus N_{A_2})$, $E_k(f_1(Ch_A), Ch_B)$
7. $A \longrightarrow B$: $E_k(f_2(Ch_B))$

Análisis

Claramente esta versión no requiere predistribución de las llaves públicas de S. A cambio, S debe generar parejas de llaves pública-privada para A y B, mantener seguras las correspondientes llaves privadas y encargarse de la distribución, también de manera segura, de las correspondientes llaves públicas de A y B. En este sentido, el protocolo agrega restricciones a la elección de los sistemas de llave pública que se utilizan porque K_A^p y K_B^p deben mantenerse en secreto entre A o B y S todo el tiempo. Se podrían olvidar K_A^p o K_A^p después de usarlas pero nunca deben comprometerse, ya que si se comprometen, entonces el mensaje 2, ($S \longrightarrow A$: A, B, N_S, $E_{P_A}(K_A^p)$, $E_{P_B}(K_B^p)$, contendría texto verificable. El protocolo toma su nombre de este modo secreto de utilizar llaves públicas.

Por razones similares, las llaves públicas K_A^p y K_B^p no deben contener redundancia conocida para un atacante. Un sistema de llave pública perfecto debe tener la propiedad de que la llave pública pueda ser cualquier número aleatorio de longitud adecuada. Además, el cifrado de mensajes aleatorios (mensajes que incluyan confundidores) no debe permitir ningún filtro de información acerca de la llave de cifrado. Hallar un sistema tal, puede resultar difícil.

Sin embargo, tradicionalmente, los criptosistemas de llave pública son evaluados contra ataques por texto en claro escogido, bajo la suposición que las llaves públicas son siempre conocidas por el atacante. Esta suposición no es cierta en este protocolo. De este modo, algunos sistemas de llave pública que previamente se consideraron desfavorables, se podrían utilizar con este protocolo.

Si se utiliza un sistema imperfecto, el hecho de usar muchas llaves K_A^p distintas puede permitir al atacante reducir el espacio de búsqueda de P_A. Afortunadamente, S puede guardar una llave pública por cada usuario, de la misma forma que guarda los passwords, y enviar siempre la misma llave pública al mismo usuario. Esta defensa tiene la ventaja de que S no necesita generar nuevas llaves públicas para cada sesión. La réplica de una llave K_A^p obsoleta o anterior provocará que A envíe un mensaje 3 inútil, pero no puede provocar rupturas de seguridad. El resto del análisis de seguridad es el mismo que para el caso *LGSN* Reducido.

Debe hacerse notar que esta alternativa de protocolo puede servir para reemplazar el *LGSN* Reducido o, también, ambos pueden implementarse en un sistema e invocar el apropiado cuando se necesite.

4.1.5 LGSN con Autenticación Directa

En algunos ambientes, no se dispone de una tercera parte segura S que actúe como servidor de llaves y certificador. En estos casos, A y B pueden ya compartir un password P, e intentar acordar de modo seguro una llave de sesión. Supóngase que K_A^p es la llave pública de A y k la llave de sesión, ambas seleccionadas por A. El protocolo *LGSN* con Autenticación Directa [1] (pp. 12 - 13) es el siguiente:

1. $A \longrightarrow B : Ch_A,\ E_P(K_A^p)$
2. $B \longrightarrow A : E_{K_A^p}(\mathrm{B},\ \mathrm{A},\ N_{B_1},\ N_{B_2},\ C_B,\ E_P(Ch_A))$
3. $A \longrightarrow B : E_P(N_{B_1},\ k \oplus N_{B_2})$
4. $B \longrightarrow A : E_k(f_1(Ch_A),\ Ch_B)$
5. $A \longrightarrow B : E_k(f_2(Ch_B))$

Análisis

Para evitar ataques por texto verificable sobre P en el mensaje 1 y porque es improbable que A recuerde llaves criptográficamente fuertes, A necesita generar una nueva llave pública K_A^p para cada sesión. Este protocolo tiene las mismas restricciones sobre la elección del sistema de llave pública que la versión para llaves públicas secretas.

Algunas veces, puede ser deseable que tanto A como B contribuyan a la elección de la llave de sesión k. Esto puede ser posible haciendo que cada parte escoja una llave k_1 y k_2 respectivamente, y usar $h(k_1, k_2)$ como llave de sesión, donde $h()$ es una función hash unidireccional.

4.1.6 LGSN Reforzado para Autenticación Directa

El protocolo *LGSN* con Autenticación Directa presentado antes asume que A y B comparten el conocimiento sobre P. Esta suposición no siempre se cumple. Lo que también es común en un esquema de passwords es que A conozca P y que B conozca y mantenga almacenado el resultado de aplicar una función hash a P, como se vió en la Subsección 1.3.1 del Capítulo 1 de este trabajo.

Una debilidad del protocolo *LGSN* con Autenticación Directa es que la llave de sesión k es calculada de forma unilateral por A. En [1] (pp. 13) se propone reforzar esa versión para permitir que tanto A como B contribuyan a la elección de k, calculando cada uno una llave k_1 y k_2 respectivamente como contribuciones propias al cálculo de la llave de sesión final, y entonces usar $k = f(k_1, k_2)$ como llave de sesión, donde $f()$ es una función hash unidireccional.

El protocolo *LGSN* Reforzado para Autenticación Directa es el siguiente:

1. A selecciona una pareja de llaves pública-privada (K_A^p, K_A^r); selecciona los núnicos N_{A_1} y N_{A_2}; selecciona el quienvive Ch_A y el confundidor C_A; calcula $h(P)$ a partir de P que conoce; cifra todos estos valores con llave $h(P)$ y los envía a B junto con su identidad:

   $A \longrightarrow B : A, E_{h(P)}(K_A^p, Ch_A, N_{A_1}, N_{A_2}, C_A)$

2. Al recibir el mensaje 1, B usa $h(P)$, que mantiene guardada, para descifrarlo y conocer K_A^p, Ch_A, N_{A_1}, N_{A_2}, y C_A. Selecciona Ch_B, utiliza $h(P)$ para cifrar Ch_A, y usa la llave pública de A, K_A^p, para cifrar el siguiente mensaje que envía a A:

   $B \longrightarrow A : E_{K_A^p}(\mathrm{B}, \mathrm{A}, N_{B_1}, N_{B_2}, C_B, Ch_B, E_{h(P)}(Ch_A))$

3. Al recibir el mensaje 2, A utiliza su llave privada K_A^r para descifrarlo, utiliza $h(P)$ para descifrar $E_{h(P)}(Ch_A)$ y verifica que Ch_A sea el que envió en el mensaje 1. Si este es el caso, selecciona la subllave k_1 y envía a B:

   $A \longrightarrow B : E_{h(P)}(N_{B_1}, k_1 \oplus N_{B_2})$

4. Al recibir el mensaje 3, B verifica que N_{B_1} y N_{B_2} sean los que envió en el mensaje 2. Si es el caso, B selecciona la subllave k_2, calcula la

llave de sesión $k = f(k_1, k_2)$, y envía a A:

$B \longrightarrow A : E_{h(P)}(N_{A_1},\ k_2 \oplus N_{A_2})$

5. Al recibir el mensaje 4, A verifica que N_{A_1} y N_{A_2} sean los que envió en el mensaje 1. Si es el caso, A calcula la llave de sesión $k = f(k_1, k_2)$, calcula $f_1(Ch_B)$, usa k para cifrar el resultado, y envía a B:

   $A \longrightarrow B : E_k(f_1(Ch_B))$

6. Al recibir el mensaje 5, B usa k para descifrarlo y calcula $f_1(Ch_B)$ para verificar que Ch_B sea el que envió en el mensaje 2. Si es el caso, calcula $f_2(Ch_A)$, usa k para cifrar el resultado y envía a A:

   $B \longrightarrow A : E_k(f_2(Ch_A))$

   Al recibir el mensaje 6, A usa k para descifrarlo y calcula $f_2(Ch_A)$ para verificar que Ch_A sea el que envió en el mensaje 1.

Análisis

Esta modificación evita que P viaje, aunque fuera cifrado de alguna forma, en alguno de los mensajes intercambiados para que $h(P)$ pueda ser calculado por B (caso $UNIX$). También, evita utilizar P como llave de cifrado en los intercambios 1 y 3 de $LGSN$ con Autenticación Directa.

Esta variante puede, a su vez, tener subvariantes como la de utilizar la llave pública de A, K_A^p, para cifrar el mensaje 4 que B envía a A, a cambio de agregar más cifrado de llave pública para obtener mayor seguridad. También se puede pensar en eliminar totalmente el cifrado de llave pública, en cuyo caso el protocolo se reduce a intercambios cifrados con llaves derivadas de P o llaves texto-equivalentes a P.

Esta variante tiene las mismas restricciones sobre la elección del sistema de llave pública que la versión para llaves públicas secretas porque sigue siendo improbable que A recuerde llaves criptográficamente fuertes, por lo que A necesita generar una nueva llave pública K_A^p para cada sesión.

4.1.7 LGSN Óptimo para Autenticación Directa

En [55] (pp. 4), L. Gong propone el siguiente protocolo óptimo para autenticación directa de dos partes:

1. A selecciona una pareja de llaves pública-privada (K_A^p, K_A^r) y el núnico N_A. Cifra la llave pública K_A^p usando como llave el password P que comparte con B y lo envía, junto con N_A, a B.

   $A \longrightarrow B : N_A,\ E_P(K_A^p)$

2. Al recibir el mensaje 1, B usa P para descifrarlo y conocer K_A^p. Selecciona aleatoriamente: el núnico N_B, el confundidor C_B, y la llave de sesión k. Cifra N_A que recibió de A usando P como llave. Después, utiliza la llave pública de A, K_A^p, para cifrar el siguiente mensaje que envía a A:

   $B \longrightarrow A : E_{K_A^p}(\mathrm{B},\ \mathrm{A},\ N_B,\ C_B,\ k,\ E_P(N_A))$

3. Al recibir el mensaje 2, A utiliza su llave privada K_A^r para descifrarlo y conocer k y N_B. Utiliza P para descifrar $E_P(N_A)$ y verifica que N_A sea el que envió en el mensaje 1. Si este es el caso, cifra N_B con k y lo envía a B:

   $A \longrightarrow B : E_k(N_B)$

Análisis

La presencia de $E_P(N_A)$ en el mensaje 2 prueba a A que el mensaje fue enviado por B y es fresco. A completa la fase del saludo enviando $E_k(N_B)$ en el último mensaje.

El análisis de seguridad es similar al del protocolo *LGSN* Original de la subsección 4.1.1 de este mismo capítulo.

Este protocolo consta de tres mensajes y tres rondas, lo cual constituye el límite inferior probado para protocolos basados en núnicos [10]. Es más eficiente que el *LGSN* Reducido de la subsección 4.1.2, el cual consta de cinco mensajes.

Resulta fácil modificar este protocolo para que ambas partes contribuyan a la elección de la llave de sesión k. Po ejemplo, A puede proponer otra llave de sesión k_1, denotar a k_2 como la llave que propone B, y entonces ambas partes usar $k = h(k_1, k_2)$ como llave de sesión final. Esta modificación dejaría el protocolo de la siguiente manera:

1. $A \longrightarrow B : N_A,\ E_P(K_A^p,\ k_1)$
2. $B \longrightarrow A : E_{K_A^p}(\mathrm{B},\ \mathrm{A},\ N_B,\ C_B,\ k_2,\ E_P(N_A))$
3. $A \longrightarrow B : E_{h(k_1,k_2)}(N_B)$

4.2 Protocolos EKE (*"Encrypted Key Exchange"*)

La familia de protocolos EKE [7, 17] utiliza una combinación de criptografía convencional y de llave pública, -usan una llave secreta (el password P) para cifrar una llave pública generada aleatoriamente- , para lograr autenticación e intercambio de llaves protegiendo los passwords contra ataques tipo diccionario fuera de línea.

Estos protocolos asumen que las partes A y B previamente a la ejecución del protocolo comparten el conocimiento sobre un password P criptográficamente débil.

La presentación y análisis de estos protocolos está basada fundamentalmente en [7, 17, 18], y para ello se usa el modelo y la notación especificados en la Sección 2.2 del Capítulo 2 de este trabajo.

El protocolo EKE original [17] (pp. 3) es el siguiente:

4.2.1 EKE Original

1. A genera una pareja de llaves K_A^p y K_A^r, pública y privada, para usarlas en un criptosistema asimétrico. Luego, A cifra K_A^p en un criptosistema simétrico usando P como llave, y la envía a B junto con su identidad en claro:

$A \longrightarrow B$: A, $E_P(K_A^p)$

2. Como A y B comparten el conocimiento sobre P, B descifra el mensaje 1 y obtiene K_A^p. Luego, genera aleatoriamente una llave de sesión k y la cifra usando el criptosistema asimétrico con llave K_A^p. El resultado lo cifra en un criptosistema simétrico usando P como llave, y la envía a B:

   $B \longrightarrow A : E_P(E_{K_A^p}(k))$

3. Como A conoce P y su llave privada K_A^r, descifra el mensaje 2 para conocer k. Luego, A genera un quienvive Ch_A y lo cifra en un criptosistema simétrico usando k como llave, y lo envía a B:

   $A \longrightarrow B : E_k(Ch_A)$

4. Como B concoce k, descifra el mensaje 3 para conocer Ch_A. Luego, genera un quienvive Ch_B, cifra Ch_A y Ch_B en un criptosistema simétrico usando k como llave, y lo envía a A:

   $B \longrightarrow A : E_k(Ch_A, Ch_B)$

5. Como A conoce k, descifra el mensaje 4 para conocer Ch_A y Ch_B. Compara Ch_A con el que envió a B. Si coincide, cifra Ch_B en un criptosistema simétrico usando k como llave, y lo envía a B:

   $A \longrightarrow B : E_k(Ch_B)$

Análisis

El análisis que se presenta de este protocolo está basado en [17, 18].

Los pasos 3 a 5, que corresponden a la parte quienvive-respuesta, son una técnica estándar para validar llaves criptográficas [17] (pp. 3). Esta parte del protocolo puede reemplazarse con otros mecanismos para validar k. Por ejemplo, puede intercambiarse una instantánea cifrada usando k como llave, bajo la suposición crítica que los relojes de A y B están sincronizados, como en *Kerberos* [4] (pp. 6).

P se usa como llave para cifrar dos veces (mensajes 1 y 2). Con frecuencia, puede omitirse una vez, dependiendo del criptosistema de llave pública que se use [17] (pp. 3).

Después de intercambiar el mensaje 2, A y B conocen: K_A^p y k. Esta última puede usarse para proteger la sesión. Por ejemplo, B podría enviar a A:

$B \longrightarrow A : E_k(Tipo\ de\ Terminal)$

Fortalezas

Considérese la posición de un atacante en esta etapa de la ejecución del protocolo. Conociendo $E_{P'}(K_A^p)$, $E_{P'}(E_{K'^p_A}(\mathrm{k}))$ y $E_k(Tipo\ de\ Terminal)$, puede conjeturar un password P' para descifrar $E_{P'}(K_A^p)$ y producir un *candidato* a llave pública $K_A'^p$. Pero, determinar si $K_A'^p$ es la llave pública usada en el intercambio es determinar si existe una llave secreta k', de tal manera que $E_{K'^p_A}(k') = E_{K^p_A}(k)$, y $D_{k'}(E_k(Tipo\ de\ Terminal))$ tengan algún sentido para el atacante.

Esta cuantificación es la *propiedad clave* del intercambio: una conjetura P' del password no puede ser rechazada sin hacer un ataque por fuerza bruta sobre k [2]. Dado que K_A^p y k son generadas aleatoriamente sobre un espacio de llaves presumiblemente grande, tales ataques son costosos aún si el espacio de passwords es pequeño. El espacio relativamente pequeño del cual se escoge P es efectivamente multiplicado por el tamaño del espacio de llaves del cual se obtiene k.

Otra manera de ver esto es examinar el resultado de intentar descifrar $K_A'^p = D_{P'}(E_P(K_A^p))$ y preguntarse si esta cantidad sería comprensible. Si K_A^p es verdaderamente aleatoria, no hay modo de decir si P' es correcto sin romper K_A^p. Debido a que K_A^p se escoge de un espacio mucho más grande que el de P, romperla es mucho más difícil [17] (pp. 2).

Debilidades

El requerimiento más importante para este protocolo es que el mensaje que se cifre usando P como llave, debe ser indistinguible de un número aleatorio. Por ejemplo, si algún criptosistema requiere usar números primos como llaves públicas, esto no serviría para cifrar el mensaje 1, porque para un atacante sería trivial verificar una conjetura de P probando la primalidad del

[2] Esta discusión asume que no se usan ataques criptoanalíticos

resultado de descifrar.

4.2.2 EKE Reforzado

Los autores de EKE [17] consideran la eventual posibilidad de un ataque usando criptoanálisis, el cual otorgaría al atacante conocimiento sobre la llave k, dándole de esta manera una vía para atacar P por diccionario. Este ataque también se conoce como *Ataque Denning-Sacco* por haber sido ilustrado por Denning y Sacco [27] en su análisis al protocolo de Needham y Schroeder [26]. El ataque se puede resumir como sigue: *el atacante manipula hasta obtener una de las llaves de sesión usadas en una ejecución del protocolo. Armado con ese conocimiento, es entonces capaz de impersonar indefinidamente a una de las partes* [18] (pp. 2).

Este ataque no se aplica directamente sobre el protocolo EKE. Sin embargo, una variación sutil del ataque es posible de aplicar: *el atacante, de alguna manera, obtiene una de las llaves distribuidas en una ejecución de EKE. Armado con este conocimiento, monta un ataque por diccionario sobre P y, una vez roto P, es capaz de impersonar indefinidamente a una de las partes.* Este ataque sobre el prrotocolo EKE se ilustra con detalle en [18] (pp. 2).

Los autores de EKE proponen una variación menor [17] (pp. 7) al protocolo para prevenir la posibilidad de este ataque. Esta variación al protocolo EKE consiste en que durante el intercambio quienvive-respuesta, A y B generen subllaves aleatorias k_A y k_B, de tal modo que el mensaje 3 se haga $E_{k_i}(Ch_A,\ k_A)$, mientras el mensaje 4 se convierte en $E_{k_i}(Ch_A,\ Ch_B,\ k_B)$, donde k_i indica que se trata de una llave inicial. Después de esto, las dos partes calculan la verdadera llave de sesión $k_s = \mathrm{f}(k_A,\ k_B)$ para alguna función f adecuada. Esta nueva llave es la que se usa para los intercambios subsecuentes; la llave inicial k_i se reduce, de esta manera, a una *llave para intercambiar llaves.*

No obstante la anterior protección, puede concebirse un ataque criptoanalítico más sofisticado que explote la presencia de quienvives y respuestas en diferentes mensajes para atacar k_i. Esto parece improbable; sin embargo, por si fuera de interés, los autores proponen modificar las respuestas (mensajes 4 y 5) para que contengan una función hash unidireccional g de los quienvives, y no los quienvives mismos [17] (pp. 7). De este modo, el mensaje 4 se hace

$E_{k_i}(g(Ch_A),\ Ch_B,\ k_B)$ y el mensaje 5 se convierte en $E_{k_i}(g(Ch_B))$.

El protocolo EKE Reforzado [18] (pp. 2) es el siguiente:

1. $A \longrightarrow B$: A, $E_P(K_A^p)$
2. $B \longrightarrow A$: $E_P(E_{K_A^p}(k_i))$
3. $A \longrightarrow B$: $E_{k_i}(Ch_A,\ k_A)$
4. $B \longrightarrow A$: $E_{k_i}(g(Ch_A),\ Ch_B,\ k_B)$
5. $A \longrightarrow B$: $E_{k_i}(g(Ch_B))$

La verdadera llave de sesión k_s es calculada entonces por las dos partes A y B como $k_s = f(k_A,\ k_B)$ para alguna función unidireccional f adecuada.

Análisis

El análisis presentado aquí, en términos de fortalezas y debilidades de este protocolo, está basado en [17, 18].

Fortalezas

Esta modificación protege aún más al protocolo EKE porque obtener una llave de sesión k_s dice menos que antes a un atacante acerca de P, ya que P nunca se usa para cifrar algo que tenga que ver directamente con k_s. Un ataque por criptoanálisis sobre k_s no es factible porque es ahora k_i la que se usa para cifrar datos aleatorios, y k_s nunca aparece en ningún mensaje.

Debilidades

En [18] (pp. 3) se menciona que, aunque poco probable, un atacante puede todavía intentar romper k_i y pretender hallar alguna información en un ataque por diccionario sobre P. Un ataque por fuerza bruta sobre k_i es posible, ya que el atacante puede iterar sobre todas las posibles elecciones de k_i y, para cada posible llave k_i', hacer:

1. Descifrar los mensajes 3 ($E_{k_i}(Ch_A, k_A)$), y 4 ($E_{k_i}(g(Ch_A), Ch_B, k_B)$), para extraer posibles valores Ch'_A y $g'(Ch_A)$

2. Calcular $g(Ch'_A)$ y compararla con la extraída.

No obstante, este ataque es difícil debido a que k_i puede ser mucho más fuerte que k_s.

Una desventaja [18] de este protocolo es su relativa complejidad. Requiere 5 mensajes y 6 operaciones de cifrado, 5 con llave secreta y 1 con llave pública, sobre un total de 8 bloques de datos. Dado que el protocolo es bastante general, hacerlo más eficiente probablemente introduzca vulnerabilidades no esperadas cuando se le adaptase a criptosistemas específicos [18] (pp. 3).

4.2.3 EKE con Diffie-Hellman (*EKE-DH*)

Como se ha visto, el protocolo *EKE* utiliza cifrado asimétrico para intercambiar una llave para un sistema de cifrado simétrico. Este es un caso donde se puede utilizar el esquema de acuerdo de llave de Diffie-Hellman [3] [39], también conocido como *intercambio exponencial de llave* [50] (pp. 515). Este intercambio no proporciona por sí mismo autenticación y, además, es vulnerable a ataques por monitoreo y hombre enmedio (Sección 2.2 del Capítulo 2). No obstante, cifrando los valores transmitidos usando P como llave, pueden resolverse ambos problemas.

La variante de *EKE* que utiliza el acuerdo de llave Diffie-Hellman se referencia como *EKE-DH* (*"Exponential Key Exchange - Diffie Hellman"*) [17] (pp. 8), y aparece como la variante más práctica debido a la relativa simplicidad del acuerdo Diffie-Hellman [18] (pp. 3).

En esta variante se asume que A y B acuerdan una base común α, y un módulo β, donde β es un número primo apropiadamente grande y α es un generador del grupo [4] multiplicativo de enteros módulo β, donde $(2 \leq \alpha \leq \beta - 2)$. Esta base y el módulo, una vez seleccionados, se hacen públicos [50] (pp. 516). Esta no es la única manera de seleccionar α y β; de hecho, existe

[3] Sección A.11 del Apéndice A
[4] Sección A.10 del Apéndice A

una variedad de posibilidades que ofrecen un amplio rango de opciones entre costo y seguridad. Una discusión interesante sobre este tema se puede hallar en [17] (pp.8 - 9).

El módulo β y el algoritmo de cifrado simétrico deben elegirse cuidadosamente para asegurarse que las exponenciaciones no contengan texto verificable para E_P [20] (pp. 2).

El tamaño considerado como adecuado en términos de seguridad para β es de 1024 bits, y para los exponentes aleatorios R_A y R_B es de 336 bits [19] (pp. 17).

El protocolo *EKE-DH* es el siguiente:

1. A selecciona un número aleatorio R_A, calcula $E_P(\alpha^{R_A}(mod\ \beta))$ y lo envía a B, junto con su identidad en claro

   $A \longrightarrow B : A,\ E_P(\alpha^{R_A}(mod\ \beta))$

2. B usa P para descifrar $E_P(\alpha^{R_A}(mod\ \beta))$, selecciona aleatoriamente un número R_B y calcula $\alpha^{R_B}(mod\ \beta)$ y $k = (\alpha^{R_A R_B})(mod\ \beta)$ que será la llave de sesión. Finalmente, B genera un quienvive, Ch_B y transmite

   $B \longrightarrow A : E_P(\alpha^{R_B}(mod\ \beta)),\ E_k(Ch_B)$

3. A usa P para descifrar $E_P(\alpha^{R_B}(mod\ \beta))$. Con este valor calcula $k = (\alpha^{R_A R_B})(mod\ \beta)$; luego, usa k para descifrar $E_k(Ch_B)$. Entonces, A genera su propio quienvive Ch_A y envía a B:

   $A \longrightarrow B : E_k(Ch_A,\ Ch_B)$

4. B descifra el mensaje 3 y verifica que Ch_B haya sido devuelto correctamente. B envía:

   $B \longrightarrow A : E_k(Ch_A)$

5. A descifra el mensaje 4, obtiene Ch_A y verifica que coincida con el original

Análisis

El análisis de esta versión del protocolo EKE se basa en [17] (pp. 7 - 8). Ya que R_A es aleatorio, también α^{R_A} lo es y por tanto las conjeturas de P no proporcionan ninguna información útil a un atacante. Como A y B conocen P pueden recuperar $\alpha^{R_B}(mod\ \beta)$ y $\alpha^{R_A}(mod\ \beta)$ respectivamente, y pueden calcular k. Un atacante no puede hacer lo mismo y por lo tanto no puede realizar un ataque por hombre enmedio. Ningún intento por conjeturar P fuera de línea es útil; aún una conjetura exitosa llevará a conocer sólo $\alpha^{R_A}(mod\ \beta)$ y $\alpha^{R_B}(mod\ \beta)$, lo cual no proporciona a un atacante ninguna información acerca de la llave de sesión k (ver Problema Diffie-Hellman en la subsección A.11.1 del Apéndice A de este trabajo).

Al usar $(\alpha^{(R_A)R_B})(mod\ \beta) = (\alpha^{(R_B)R_A})(mod\ \beta)$ como llave, un atacante que conozca sólo $\alpha^{R_A}(mod\ \beta)$ y $\alpha^{R_B}(mod\ \beta)$, no puede realizar el mismo cálculo. Además, ninguna parte puede controlar la elección de k; hacer eso, es equivalente a resolver el problema de logaritmos discretos [5] [29].

Debido a que el proceso de intercambio de la llave en sí mismo produce una llave de sesión aleatoria, no se requiere una transmisión separada de k.

En [19] (pp. 6) se detallan algunas restricciones sobre $EKE\text{-}DH$ para evitar ataques criptoanalíticos por: logaritmos discretos, partición, y confinamiento de k a un subgrupo [6].

$EKE\text{-}DH$ es resistente al ataque Denning-Sacco mostrado sobre EKE en [18] (pp. 2). Su resistencia a este ataque se debe en gran parte al hecho de que k nunca se intercambia. Sólo los residuos $(mod\ \beta)$ se transfieren en forma cifrada. Aún si el atacante obtiene ambos residuos $\alpha^{R_A}(mod\ \beta)$ y $\alpha^{R_B}(mod\ \beta)$, no puede acercarse al descubrimiento de k. Inversamente, si el atacante de algún modo descubre k, no puede validar la correcta conjetura de $\alpha^{R_A}(mod\ \beta)$ y $\alpha^{R_B}(mod\ \beta)$, haciendo imposible de esta manera un ataque por diccionario, por lo que no requiere el reforzamiento mostrado para EKE en la subsección anterior [18] (pp. 3).

En [17, 18] y [38] (pp. 518 - 522) se sugiere la omisión de al menos un cifrado simétrico en los intercambios 1 y 2. En [17] (pp. 8) los autores sugieren

[5] Subsecciones A.11.1 y A.11.2 del Apéndice A
[6] Subsección A.10.2 del Apéndice A

omitir el cifrado en el intercambio 1, de tal manera que este intercambio se convierta en

$A \longrightarrow B : A,\ \alpha^{R_A}(mod\ \beta)$

en lugar de

$A \longrightarrow B : A,\ E_P(\alpha^{R_A}(mod\ \beta))$

Con esta modificación, un atacante aún no es capaz de descifrar la respuesta de B, y por tanto no obtendrá ninguna información. Ningún ataque activo que intente sustituir $\alpha^{R_A}(mod\ \beta)$ tendrá éxito; el atacante no puede responder al quienvive en el intercambio 2 sin conocer el verdadero valor de R_A.

Omitir algún cifrado en los intercambios 1 y 2 es, no obstante, muy delicado porque puede abrir la posibilidad de que el protocolo pueda ser vulnerable a ataques por texto escogido o por diccionario. Por ejemplo, un análisis de *EKE-DH* hecho en [18] (pp. 3 - 4) muestra que si se elimina el cifrado del mensaje 2, resulta que el protocolo se hace vulnerable a ataques por diccionario. Esto se debe a que enviar $\alpha^{R_B}(mod\ \beta)$ en claro, daría al atacante la oportunidad de seleccionar R_B y el correspondiente $\alpha^{R_B}(mod\ \beta)$. Todo lo que tiene que hacer el atacante es hacerse pasar por B en el mensaje 2. En el mensaje 3, recibe $E_k(Ch_A, Ch_B)$ de A. Entonces, para cada conjetura de P, el atacante hace lo siguiente:

1. Descifra $E_P(\alpha^{R_A}(mod\ \beta))$ del mensaje 1 y construye un candidato a llave de sesión k'
2. Descifra $E_k(Ch_B)$ (el cual el mismo atacante urdió en el mensaje 2) con k', y obtiene $D_{k'}(E_k(Ch_B))$
3. Descifra $E_k(Ch_A, Ch_B)$ (recibido de A en el mensaje 3), y obtiene $D_{k'}(E_k(Ch_A, Ch_B))$
4. Si la segunda mitad de $D_k(E_k(Ch_A, Ch_B))$ coincide con $D_{k'}(E_k(Ch_B))$, el password conjeturado es correcto.

La anterior vulnerabilidad es fácil de corregir arreglando ligeramente los mensajes del protocolo como sigue:

1. $A \longrightarrow B : A,\ E_P(\alpha^{R_A}(mod\ \beta)),\ Ch_A$

2. $B \longrightarrow A : (\alpha^{R_B}(mod\ \beta)),\ E_k(Ch_B,\ Ch_A)$

3. $A \longrightarrow B : E_k(Ch_B)$

Ya que A puede autenticar a B después del mensaje 2, entonces puede detener la ejecución del protocolo en caso de que el quienvive Ch_A no coincida, y por lo tanto evitar el ataque.

En [19] (pp. 10 - 11) se presenta la descripción del ataque por diccionario sobre *EKE-DH* cuando se omite el cifrado en el mensaje 1, y el atacante I ahora suplanta a A. Se resalta que este escenario de ataque se debe al hecho de que en la ejecución del protocolo, B muestra su conocimiento de k antes de que I (suplantando a A) muestre el suyo. Este escenario de ataque se corrige si se cambia el orden de verificación, de tal manera que la parte que primero recibe la prueba no sea la misma parte que envía el resultado de la exponenciación sin cifrar.

4.2.4 EKE Mínimo con Diffie-Hellman (*MEKE*)

Una de las ventajas de usar un acuerdo de llaves Diffie-Hellman para el protocolo *EKE-DH* es su inherente democracia ya que ambas partes contribuyen por igual a la llave resultante. Este hecho y su relativa simplicidad en implementación lo hacen atractivo y práctico para ambientes tales como tarjetas inteligentes y calculadoras criptográficas. En tales ámbitos un problema típico es la eficiencia, la cual a su vez tiene que ver con el número de intercambios, rondas, y operaciones de cifrado del protocolo, entre otros conceptos.

A partir de la suposición que todas las unidades de datos - nombres, residuos, quienvives y sus correspondientes cifrados - son de la misma longitud, *EKE-DH* involucra siete bloques de datos. Un protocolo de dos partes basado en quienvives requiere un mínimo de tres mensajes y tres rondas [10]. Se puede demostrar que el mínimo absoluto en bloques de datos es de cinco [18] (pp. 4).

A la luz de las consideraciones anteriores, en [18] (pp. 4 - 5) se presenta una

versión de *EKE-DH* que eficientiza el protocolo en términos de número de mensajes, rondas y bloques de datos.

La versión de *EKE* Minimizado (*MEKE*) es la siguiente:

1. $A \longrightarrow B : A,\ E_P(\alpha^{R_A}(mod\ \beta))$
2. $B \longrightarrow A : \alpha^{R_B}(mod\ \beta),\ E_k(\alpha^{R_B}(mod\ \beta))$
3. $A \longrightarrow B : E_k(\mathrm{f}(\alpha^{R_B}(mod\ \beta)))$

Análisis

Esta versión de *EKE-DH* minimizada consta de tres mensajes y tres rondas. Requiere tres operaciones de cifrado: una con P y dos con k; las tres se realizan sobre un único bloque de datos. Durante la ejecución del protocolo se intercambian en total cinco bloques de datos. A y B generan sólo un número aleatorio cada uno [18] (pp. 5).

Otra ventaja de *MEKE* es que no requiere fortaleza adicional contra ataques por criptoanálisis, ya que el conocimiento de una llave de sesión k no abre el camino para atacar por diccionario a P, o para atacar otras llaves de sesión. Lo único que se necesita reforzar es el cálculo de la llave de sesión k: en lugar de calcular $k = (\alpha^{R_A R_B})(mod\ \beta)$ como en *EKE-DH*, se calcula $k = h((\alpha^{R_A R_B})(mod\ \beta))$. Los ataques por diccionario no son posibles porque aún cuando el atacante descubra k, no es capaz de calcular ninguna información sobre $(\alpha^{R_A})(mod\ \beta)$ a partir de k y de $(\alpha^{R_B})(mod\ \beta)$. Para el atacante, $(\alpha^{R_A})(mod\ \beta)$ es un valor aleatorio y el conocer el mensaje 1 no posibilita al atacante para atacar P [18] (pp. 9).

Los ataques activos no son posibles ya que aún cuando el atacante construya el primer mensaje, B siempre producirá y enviará un segundo mensaje. Construir el segundo mensaje enviado a A no significa ninguna ventaja sobre un ataque por diccionario [19] (pp. 9).

Un ataque por texto verificable a partir de los mensajes 2 y 3 no parece factible sobre *MEKE* debido a la imposibilidad, para un atacante, de calcular k a partir del conocimiento de $(\alpha^{R_B})(mod\ \beta)$.

4.2.5 AEKE (*"Augmented Encrypted Key Exchange"*)

El protocolo EKE requiere que A y B tengan versiones en claro de P, una restricción que no siempre puede cumplirse.

En [7] se presenta una extensión del protocolo EKE para manejar la situación cuando B guarda el resultado de aplicar alguna función hash unidireccional $h(P)$ a P, y la validación de P se realiza calculando $h(P')$ del password P' tecleado, y comparando el resultado con el valor almacenado.

El usar $h(P)$ por parte de A y B evita la necesidad de que B tenga que conocer P en claro, pero, un ladrón que roba $h(P)$ de B puede usarlo para hacerse pasar por A ante B. Una solución a este problema es el protocolo $AEKE$. En este método el verificador en B es $h(P)$, el cual se usa para verificar una prueba de que A conoce P. El ladrón no puede usar $h(P)$ directamente para hacerse pasar por A.

Esta extensión llamada $AEKE$ asume que el resultado hash $h(P)$ se guarda en B pero que ambas partes usan $h(P)$, el cual debe ser secreto. No obstante, debido a que se asume que bajo alguna circunstancia $h(P)$ puede comprometerse, A debe enviar un mensaje adicional conteniendo una diferente función hash unidireccional de P; este valor, junto con $h(P)$ y la llave de sesión, es usado por B para validar la secuencia de *"login"*.

Debe quedar claro que esta variación de EKE no evita que un atacante que obtenga $h(P)$ pueda suplantar a B y montar un ataque diccionario sobre P. Lo que trata de garantizar es que el atacante no pueda suplantar a A ante B. Por esto el requerimiento de que $h(P)$ debe ser secreto.

Los objetivos de este protocolo son los mismos que los de EKE, preservando las propiedades de este último. Brinda protección a los passwords contra ataques diccionario fuera de línea y contra compromisos del archivo donde se guarda $h(P)$, llamado comúnmente archivo de passwords. Esto último significa evitar que un atacante que obtiene, de alguna manera de B, el archivo que contiene los resultados hash de los passwords no pueda conocer P ni suplantar a A.

Dos maneras de lograr estos objetivos son: una por medio del uso de firmas

digitales [7], la cual se mostrará en esta sección, y la otra basándose en una familia de funciones hash unidireccionales conmutativas [7] (pp. 4).

A continuación se mostrará la versión de EKE que maneja $h(P)$ en lugar de P en claro, como lo hace EKE original. Posteriormente se mostrará la versión de $AEKE$ con firmas digitales.

EKE con Resultados Hash (AEKE)

Esta extensión al protocolo EKE requiere algunas precisiones. Primero, $h(P)$ como resultado hash unidireccional de P no permite la factibilidad de recuperar P a partir de $h(P)$. Segundo, A y B deben intentar mantener en secreto $h(P)$, pero esto no puede garantizarse. El objetivo es evitar que un atacante que capture $h(P)$ pueda conocer P, o suplantar a A. Además, un atacante que no tiene $h(P)$ es incapaz de montar un ataque por diccionario contra P. Tercero, $f(P,k)$ es un resultado hash que depende de P y de la llave de sesión k. A calcula esta cantidad, la cifra con la llave k y envía $E_k(f(P,k))$ a B. Tercero, se define un predicado $T(h(P), f(P,k), k)$ que se evalúa verdadero si y solo si el P genuino fue usado para crear $h(P)$ y $f(P,k)$. Es decir, $T(X,Y,Z)$ es verdadero si y solo si $X = h(P)$ y $Y = f(P,Z)$, para el verdadero P.

El protocolo EKE con Resultados Hash ($AEKE$) [7] (pp. 3) es el siguiente:

1. A calcula $h(P)$, selecciona un número aleatorio R_A, calcula $E_{h(P)}(\alpha^{R_A} (mod\ \beta))$ y lo envía a B, junto con su identidad en claro

   $A \longrightarrow B : A,\ E_{h(P)}(\alpha^{R_A}\ (mod\ \beta))$

2. B selecciona aleatoriamente un número R_B y utiliza $h(P)$ para descifrar $E_{h(P)}(\alpha^{R_A} (mod\ \beta))$. Calcula la llave de sesión $k = (\alpha^{R_A R_B})(mod\ \beta)$. Finalmente, B genera un quienvive, Ch_B, y transmite

   $B \longrightarrow A : E_{h(P)}(\alpha^{R_B} (mod\ \beta)),\ E_k(Ch_B)$

3. A usa $h(P)$ para descifrar $E_{h(P)}(\alpha^{R_B} (mod\ \beta))$. De este valor, calcula k, el cual usa para descifrar $E_k(Ch_B)$. Entonces, A genera su propio quienvive Ch_A y envía a B:

[7] Sección A.5 del Apéndice A

$A \longrightarrow B : E_k(Ch_A, Ch_B)$

4. B descifra el mensaje 3 y verifica que Ch_B haya sido devuelto correctamente. B envía:

   $B \longrightarrow A : E_k(Ch_A)$

5. A descifra el mensaje 4, obtiene Ch_A y verifica que coincida con el original

6. En el paso anterior termina la ejecución de EKE usando $h(P)$ en lugar de P en claro. Ahora, la idea es que un atacante que haya obtenido $h(P)$, no sea capaz de engañar al servidor haciéndose pasar por A. Por esto, el protocolo se extiende un paso más. A envía a B:

   $A \longrightarrow B : E_k(f(P, k))$

7. Al recibir el mensaje, B lo descifra usando k para obtener $f(P, k)$ el cual utiliza, junto con $h(P)$ que conoce, para evaluar el predicado. El protocolo termina exitosamente sólo si el predicado $T(h(P), f(P, k), k)$ se evalúa a verdadero.

Análisis

El análisis de este protocolo se basa en [7] (pp. 2 - 3). Debe notarse que un atacante que pueda obtener $h(P)$ de B puede suplantarlo ante A; pero sin conocer P, no puede suplantar a A ante B.

Los cinco primeros pasos de $AEKE$, son simplemente los de EKE con $h(P)$ en lugar de P. La idea básica de estos pasos es ejecutar primero EKE con $h(P)$ en lugar de P en claro. El resultado es una llave de sesión k que debería ser conocida sólo por A y B, y en la práctica sólo puede ser conocida por alguien que conozca $h(P)$. Hasta el intercambio 5, quien conozca $h(P)$, será capaz de hacerse pasar por B, por A, o por ambos, y será capaz de montar ataques diccionario.

Consecuentemente, A debe proporcionar $f(P, k)$ para persuadir a B de su identidad. El hecho de enviar P en claro, o como $E_k(P)$, es ciertamente inseguro contra un atacante que estuviera suplantando al genuino B.

En [18] (pp. 2) se menciona que si un atacante puede obtener k de alguna forma, entonces puede montar un ataque diccionario sobre P usando

precisamente el último mensaje intercambiado, $(E_k(f(P,k)))$.

El protocolo puede reforzarse contra ataques tipo diccionario haciendo que B genere un valor s aleatorio y guarde la pareja $(h(s,P),s)$ en lugar de $h(P)$. B podría entonces enviar s en claro a A como paso inicial adicional del protocolo EKE, y $h(s,P)$ sería usado en lugar de $h(P)$. Esto proporciona dos de las tres ventajas de la sal de UNIX [2, 13], las cuales son: es imposible decir si dos resultados hash de P usan el mismo texto en claro y, es imposible construir un diccionario de conjeturas de passwords a los que previamente se les haya calculado el hash. Esto se debe al gran número de resultados hash posibles que agrega la sal para cada password.

No obstante, el cambio anterior permite a un atacante un grado adicional de libertad sobre el protocolo, el cual debe ser considerado cuidadosamente en el diseño de una implementación segura. Puede haber varias maneras seguras de escoger $h()$, $f()$, y $T()$ que satisfagan los requerimientos del protocolo. Entre estas maneras están: firmas digitales y funciones conmutativas unidireccionales [7] (pp. 3). En la siguiente sección se presenta el caso de $AEKE$ con firmas digitales.

AEKE con Firmas Digitales

Conceptualmente, un modo directo de implementar $AEKE$ es definir $h(P)$ *como llave pública de A en un esquema de firma digital* [8] [7] (pp. 3). Para calcular $f(P,k)$, A firma k con su llave privada K_A^r. Al momento de registrar a A (usuario), B (servidor) mantiene solamente la llave pública de A, es decir, $h(P)$. Al momento de *"login"*, A utiliza su llave pública $h(P)$ para el intercambio $EKE\text{-}DH$, y su llave privada para la extensión $AEKE$ [7] (pp. 3).

En esquemas de firma digital como ElGamal [29], calcular la llave pública puede ser mucho más costoso que calcular la llave privada. Cualquier número, como P o una función de él, puede ser una llave privada, pero la llave pública debe calcularse. De este modo, puede ser más eficiente para B guardar ambos: $h(P)$ y una llave pública $K^p(P)$ derivada de P. Entonces puede usarse $h(P)$, que es fácil de calcular, para el intercambio $EKE-DH$, y la llave pública puede usarse para verificar la firma. Además, la generación

[8] Sección A.5 del Apéndice A

de la llave privada $K^r(P)$, que puede ser P o derivarse de P, puede hacerse en parelelo con el intercambio *EKE-DH* [7] (pp. 3).

En esta versión de *AEKE* la llave pública para el algoritmo de firma digital K_A^p es $h(P)$ y es mantenida por B. A calcula K_A^p a partir de P. Debe notarse cuidadosamente la distinción entre la operación de firma digital $S_{K^p}(X, M)$ en un criptosistema asimétrico que involucra la firma X del mensaje M con llave pública K^p, y $E_{K^p}(X)$ como el cifrado de X en un criptosistema simétrico usando la llave K^p de un sistema de firma digital [7] (pp. 3).

El protocolo *AEKE* con Firmas Digitales, inicia con un intercambio *EKE-DH* del cual se deriva k basándose en el conocimiento compartido de A y B sobre $h(P)$ que en este protocolo tiene dos funciones: como llave pública de A, y como llave secreta que B almacena y que A calcula. Después de esto, A firma k con $K_A^r = P$ o alguna llave privada derivada de P, cifra la firma usando k como llave simétrica y la envía a B. Luego, B descifra la firma usando k, verifica la firma usando $K_A^p = h(P)$ y, si la firma es correcta, B confirma que A conoce P.

El protocolo *AEKE* con Firmas Digitales [7] (pp. 4) es el siguiente:

1. A selecciona un número aleatorio R_A, calcula $E_{h(P)}(\alpha^{R_A}(mod\ \beta))$ y lo envía a B, junto con su identidad en claro

   $A \longrightarrow B : A,\ E_{h(P)}(\alpha^{R_A}(mod\ \beta))$

2. B selecciona un número aleatorio R_B y utiliza la llave pública de A compartida para firma digital $K_A^p = h(P)$ para descifrar (vía un criptosistema simétrico) $E_{h(P)}(\alpha^{R_A}(mod\ \beta))$. Calcula la llave de sesión $k = (\alpha^{R_A R_B})(mod\ \beta)$. Finalmente, B genera un quienvive Ch_B y transmite

   $B \longrightarrow A : E_{h(P)}(\alpha^{R_B}(mod\ \beta)),\ E_k(Ch_B)$

3. A usa $h(P)$ para descifrar $E_{h(P)}(\alpha^{R_B}(mod\ \beta))$. De este valor, calcula k, que utiliza para descifrar $E_k(Ch_B)$. Entonces, A genera su propio quienvive Ch_A y envía a B:

   $A \longrightarrow B : E_k(Ch_A,\ Ch_B)$

4. B descifra el mensaje 3 y verifica que Ch_B haya sido devuelto correctamente. B envía:

$B \longrightarrow A : E_k(Ch_A)$

5. A descifra el mensaje 4, obtiene Ch_A y verifica que coincida con el original. A firma k usando $K_A^r = P$ o alguna llave privada derivada de P; cifra la firma usando k como llave simétrica y la envía a B:
   $A \longrightarrow B : E_k(S_{K_A^r}(k))$
6. Al recibir el mensaje 5, B lo descifra usando k para obtener $S_{K_A^r}(k)$, verifica la firma usando $K_A^p = h(P)$ y B se convence que A conoce P solo si la verificación $V_{K_A^p}(D_k(E_k(S_{K_A^r}(k))), k)$ es verdadera (Sección A.5 del Apéndice A).

Análisis

El análisis de este protocolo se basa en [7] (pp. 4 - 6).

AEKE es esencialmente la suma de un método de firma digital, que se puede denotar como "A", para extender *EKE-DH*. Este método "A" usa P como llave privada para un verificador de "llave pública secreta", que en este caso es $h(P)$, guardado por B. A pesar de que un atacante que tenga en su poder el verificador $h(P)$ puede romper P, esto puede evitarse si este verificador se mantiene en secreto o P es criptográficamente fuerte. Si esto último se cumple, entonces la fortaleza que se agrega es que un verificador robado no se puede usar directamente para suplantar a A ante B.

En el protocolo *AEKE*, la versión *EKE-DH* se utiliza para negociar k, basado en el conocimiento compartido de un verificador secreto $h(P)$. En la versión de *AEKE* con firmas digitales, las llaves privada-pública ($K_A^r(P)$, $K_A^p(P)$) para firma digital se escogen como función de P siendo $h(P)$ y P respectivamente esta pareja de llaves. B guarda la llave pública $K_A^p(P)$ como segundo verificador para P [20] (pp. 2).

Implícito en la descripción de esta variante de *AEKE* está que las transformaciones que la integran no "filtran" ninguna información útil a un atacante potencial. Es decir, es computacionalmente infactible para un atacante activo o pasivo obtener información útil acerca de P, o engañar a cualquier participante legítimo del protocolo, por réplica o por otra estrategia [7] (pp. 4).

Los autores analizan la seguridad de esta variante de *AEKE* a partir de dos

enfoques complementarios para formalizar los requerimientos de seguridad: el primero postula que las identidades algebraicas requeridas por el protocolo son las únicas identidades que cumplen las componentes de los criptosistemas y las transformaciones de los mensajes; el segundo es un enfoque más pragmático y se refiere al examen de las implementaciones concretas, para conocer o descubrir fallas de seguridad.

El protocolo $AEKE$ y sus variantes están diseñados para resistir dos distintos tipos de ataques: primero, un atacante sin ningún conocimiento de $h(P)$, no cuenta con suficiente información para montar un ataque diccionario contra P o $h(P)$; segundo, un atacante que conozca $h(P)$ no es capaz de suplantar a A ante B, ni es capaz de obtener información útil acerca de P (sin hacer un ataque diccionario). Detalles de este análisis de seguridad puede verse en [7] (pp. 4 - 6).

Una implementación alterna del protocolo EKE original se basa en criptosistemas de llave pública, en lugar de intercambio de llave exponencial. Esta alternativa tiene la desventaja de que k es seleccionada por una de las partes, en oposición a la variante con intercambio exponencial de llave, donde k es generada con la colaboración de las partes.

Si esta implementación alterna se usa con $AEKE$, el protocolo falla en presencia de un atacante activo I si A o B seleccionan k. Si A escoge k, el atacante puede imponer k a B, y eso permite el uso de $E_k(f(P,k))$, legítimamente generada, para autenticar a I ante B. Si B escoge k, el atacante impone k a A, obteniendo otra vez $E_k(f(P,k))$ para hacerse pasar por B ante A [7] (pp. 6).

4.3 Protocolo SPEKE (*"Simple Password Exponential Key Exchange"*)

Al igual que los protocolos $EKE\text{-}DH$, $MEKE$, y $AEKE$ de las secciones anteriores, el protocolo $SPEKE$ [19] se basa en un acuerdo de llaves Diffie-Hellman [39]. Como también se mencionó antes, un intercambio Diffie-Hellman por sí mismo no proporciona autenticación, y es vulnerable a ataques tipo hombre enmedio [17, 19]. En consecuencia con lo anterior, $SPEKE$ utiliza Diffie-Hellman para proteger a P de ataques tipo diccionario, y usa

P para evitar ataques tipo hombre enmedio [19] (pp. 4).

El protocolo $SPEKE$ se relaciona de manera muy cercana con EKE-DH; pero, no obstante su similitud, las restricciones preventivas de seguridad sobre ellos son diferentes.

El objetivo específico de este protocolo es proporcionar autenticación mutua y establecimiento de llave, usando sólo un password débil y previniendo ataques tipo diccionario fuera de línea.

4.3.1 SPEKE

La presentación y análisis de este protocolo está basado en [19] (pp. 4 - 13) y en el modelo y la notación especificados en la Sección 2.2 del Capítulo 2 de este trabajo.

En $SPEKE$, antes de cualquier intercambio, A y B acuerdan el uso de P y una función F para determinar los parámetros para el establecimiento de llave Diffie-Hellman. También acuerdan previamente una función hash $h()$ que se usará para calcular k. $SPEKE$ tiene dos etapas. La primera etapa utiliza intercambio Diffie-Hellman para establecer una llave de sesión compartida k, pero en lugar de la comúnmente usada base primitiva fija α, la función F convierte a P en una base $F(P)$ para exponenciación módulo un número primo grande p, elegido de tal forma que $p-1$ tenga un factor primo grande q. El resto de la primera etapa es puramente Diffie-Hellman, donde A y B inician escogiendo dos números aleatorios R_A y R_B.

El protocolo $SPEKE$ es el siguiente:

1. A selecciona un número aleatorio R_A, calcula $Q_A = F(P)^{R_A} \ (mod\ p)$ y lo envía a B. La identidad de A, como en muchos protocolos, se desprende del contexto. Por esta razón no se envía a B, pero puede incluirse si se desea

   $A \longrightarrow B : Q_A = F(P)^{R_A} \ (mod\ p)$

2. B selecciona aleatoriamente un número R_B, calcula $Q_B = F(P)^{R_B}$ $(mod\ p)$ y lo envía a A:

$B \longrightarrow A : Q_B = F(P)^{R_B} \ (mod\ p)$

3. A calcula $k = h(Q_B^{R_A} \ (mod\ p))$
4. B calcula $k = h(Q_A^{R_B} \ (mod\ p))$

En la segunda etapa de *SPEKE*, A y B se confirman mutuamente el conocimiento de k, antes de usarla como llave de sesión. Un modo de hacer esto es:

5. A selecciona un quienvive Ch_A, calcula $E_k(Ch_A)$ y lo envía a B:
   $A \longrightarrow B : E_k(Ch_A)$
6. B selecciona un quienvive Ch_B, calcula $E_k(Ch_B, Ch_A)$ y lo envía a A:
   $B \longrightarrow A : E_k(Ch_B, Ch_A)$
7. A descifra el mensaje 6, verifica que Ch_A sea correcto y envía a B:
   $A \longrightarrow B : E_k(Ch_B)$
8. B descifra el mensaje 7 y verifica que Ch_B sea correcto.

Análisis

La función F se escoge para crear una base de orden primo grande. En la segunda etapa del protocolo, la de confirmación de k, pueden utilizarse otras variaciones para llevarla a cabo. La que se presenta aquí es idéntica a la utilizada en el protocolo *EKE-DH*. De manera general, la etapa de verificación de k puede usar cualquier método clásico, ya que k es criptográficamente fuerte. Una forma alterna de hacerlo, usando k en lugar de quienvives, es la siguiente:

1. A envía a B su prueba de que conoce k:
   $A \longrightarrow B : h(h(k))$
2. B verifica que $h(h(k))$ es correcta y devuelve a A
   $B \longrightarrow A : h(k)$
3. A verifica que $h(k)$ es correcta

Esta variación es posible porque k se construye usando información aleatoria tanto de A como de B.

El análisis de seguridad de $SPEKE$ se basa fundamentalmente en aspectos criptoanalíticos, razón por la cual no será tratada en este trabajo. El detalle del análisis de este protocolo sobre ataques por logaritmos discretos [9], cálculos de logaritmos Pohlig-Hellman, partición, confinamiento a un subgrupo de k, y otros, puede verse en [19] (pp. 6 - 16).

4.4 Protocolo SPEKE Restringido

En [19] (pp. 7 - 16) se analiza el protocolo $SPEKE$ a la luz de ataques por criptoanálisis no considerados en este trabajo. No obstante, a pesar de la complejidad de este tipo de análisis, las restricciones derivadas de él se pueden plasmar de manera sencilla en una nueva versión de $SPEKE$ que se llamará restringida en el sentido de su fortalecimiento para soportar ataques criptoanalíticos por logaritmos discretos, partición, y confinamiento de la llave k a un subgrupo de Z_p^*, con primo p, donde $p = 2q + 1$ para un primo q.

Esta versión restringida de $SPEKE$ se describe en [19] (pp. 16 - 17) y asume el conocimiento de un número primo seguro p. Seguro en el sentido de no producir los pequeños subrupos [10] G_1 y G_2. El protocolo es el siguiente:

4.4.1 SPEKE Restringido

En la primera etapa:

1. A selecciona un número aleatorio R_A, calcula $Q_A = (P)^{2R_A} \ (mod\ p)$ y lo envía a B:

   $A \longrightarrow B : Q_A = (P)^{2R_A} \ (mod\ p)$

[9] Subsección A.11.2 del Apéndice A

[10] Sección A.10 del Apéndice A

2. B selecciona aleatoriamente un número R_B, calcula $Q_B = (P)^{2R_B}$ $(mod\ p)$ y lo envía a A:

   $B \longrightarrow A : Q_B = (P)^{2R_B}\ (mod\ p)$

3. A calcula $k_1 = (Q_B)^{2R_A}\ (mod\ p)$

4. B calcula $k_1 = (Q_A)^{2R_B}\ (mod\ p)$

Cada una de las partes aborta la ejecución si $k_1 < 2$. La idea de esta prueba sobre k_1 es evitar el confinamiento de k_1 dentro de un pequeño y predecible conjunto de valores, provocado por el uso ya sea de A, de B o de ambas, de un número t de pequeño orden como base para la exponenciación. Esta prueba trata de evitar el ataque conocido como *ataque por confinamiento a un subgrupo*. Detalles sobre este ataque se describen en [19] (pp. 9 - 10).

En la segunda etapa, A y B se confirman mutuamente el conocimiento de k_1, y derivan de ese conocimiento la llave de sesión k.

1. B calcula $V_B = h(h(h(k_1)))$, y envía el resultado a A:

   $B \longrightarrow A : V_B = h(h(h(k_1)))$

2. A verifica V_B. Si es correcto, calcula $V_A = h(h(k_1))$, y envía el resultado a B:

   $A \longrightarrow B : V_A = h(h(k_1))$

3. B verifica V_A. Si es correcto, ambas partes calculan la llave de sesión autenticada $k = h(k_1)$, y destruyen todas las copias de k_1, R_A, R_B.

Análisis

Esta versión de $SPEKE$ muestra la estructura sencilla de la versión restringida. La descripción supone la elección de un número primo p conocido y seguro (que no produzca subgrupos pequeños [11]), y de los números aleatorios R_A y R_B de tamaño adecuado. Respecto al tamaño adecuado, de acuerdo a los tamaños usados para DSA (*"Digital Signature Algorithm"*) establecidos en [66], utilizando de 512 a 1024 bits para p y 160 bits para los exponentes

[11] Subsección A.10.2 del Apéndice A

aleatorios, da como resultado una k de 180 bits, que puede ser suficiente para propósitos como los de $SPEKE$ [19] (pp. 17).

También, es posible formular de otras maneras esta versión. Un ejemplo sencillo sería tomar $F(P)$ en lugar de P.

El 2 en el exponente óbliga a la exponencial a ser un generador del subgrupo de orden q, y la k resultante es probada para asegurarse que no es 1.

4.5 Protocolos Extendidos

Buscando los mismos objetivos que las familias de protocolos EKE y $SPEKE$, en [20] se extienden los protocolos $SPEKE$, $AEKE$ y $EKE\text{-}DH$ para obtener nuevos métodos extendidos conocidos como $A\text{-}SPEKE$, $B\text{-}SPEKE$, y $B\text{-}EKE$ [20] (pp. 3). La nomenclatura obedece a la facilidad de clasificación a partir del hecho de que ya existe un protocolo $A\text{-}EKE$, o $AEKE$, presentado en la subsección 4.2.5 de este mismo capítulo.

Estos métodos extendidos buscan mejorar el rendimiento de EKE y $SPEKE$ haciendo de P un factor independiente en el proceso de autenticación.

El protocolo $A\text{-}SPEKE$ resulta de una aplicación directa de $AEKE$ a $SPEKE$. El protocolo $B\text{-}SPEKE$ es similar a $A\text{-}SPEKE$ en que usa técnicas de llave pública, pero utiliza un segundo intercambio Diffie-Hellman, en lugar de firma digital, para probar el conocimiento de A sobre P. $B\text{-}SPEKE$ asume que A y B tienen acceso al verificador $h(P)$, y que sólo A conoce P [20] (pp. 3).

El protocolo $B\text{-}EKE$ resulta de reemplazar $SPEKE$ con $EKE\text{-}DH$ en el protocolo $B\text{-}SPEKE$ [20] (pp. 3). Los protocolos $AEKE$ y $SPEKE$ se describieron y analizaron en secciones previas de este capítulo.

4.5.1 B-SPEKE

En esta subsección se mostrará, como ejemplo de estos métodos extendidos, el protocolo $B\text{-}SPEKE$. Este protocolo usa técnicas de llave pública, pero utiliza un segundo intercambio Diffie-Hellman [39] en lugar de firma digital, como lo hace la versión $AEKE$ con firmas digitales, para probar que A conoce P.

El protocolo $B\text{-}SPEKE$ asume que ambas partes tienen acceso a un verificador $V_1 = g^P \ (mod\ p)$, donde g es una base apropiada para el acuerdo de llave Diffie-Hellman y p, como antes, es un número primo grande conocido y seguro. También se asume que sólo A conoce P. Los elementos que actúan como verificadores son g, V_1, $h(P)$, los cuales se almacenan en B.

Para autenticación mutua, el intercambio primario $SPEKE$ usa $h(P)$ para crear una llave compartida k_1. A calcula $h(P)$ y B la mantiene almacenada. En el intercambio Diffie-Hellman secundario, B selecciona un exponente temporal aleatorio X, y envía $Ch = (g)^X$ como un quienvive. A calcula $k_2 = (Ch)^P$, y B calcula $k_2 = h(P)^X$, dando como resultado una segunda llave compartida. A prueba entonces el conocimiento de los valores combinados k_1, k_2 a B, probando de esta manera su conocimiento de P.

El protocolo extendido B-$SPEKE$ es el siguiente:

1. A envía su identidad a B:

   $A \longrightarrow B : A$

2. B selecciona aleatoriamente un número R_B, escoge g, calcula $Q_B = (h(P))^{2R_B}$, y envía a B:

   $B \longrightarrow A : g, Q_B$

3. A selecciona aleatoriamente un número R_A, calcula $Q_A = (h(P))^{2R_A}$, calcula una primera llave $k_1 = (h(P))^{2R_B R_A}$ y envía a B el valor Q_A y una función de prueba fp sobre A y k_1, cuyo objetivo es probar a B que se trata de A y que conoce la k_1 que B calculará (ver sección de análisis de este protocolo):

   $A \longrightarrow B : Q_A, fp(A, k_1)$

4. B calcula $k_1 = (h(P))^{2R_A R_B}$ y envía a A la función de prueba fp, sobre B y k_1:

   $B \longrightarrow A : fp(B, k_1)$

   En la segunda etapa, B genera un número aleatorio X, calcula $Ch = g^X (mod\ p)$ y lo envía a A:

   $B \longrightarrow A : Ch$

5. A calcula $k_2 = g^{XP} (mod\ p)$ y envía a B la función de prueba fp, sobre k_1 y k_2:

   $A \longrightarrow B : fp(k_1, k_2)$

6. B calcula $k_2 = V_1^X = g^{XP}$ $(mod\ p)$, que una vez verificada a partir de $fp(k_1, k_2)$, se convierte en la llave de sesión final que utilizarán A y B para comunicarse de manera segura.

Análisis

El análisis de seguridad que se resume en esta sección puede verse con mayor detalle en [20] (pp. 3 - 6).

La seguridad de *B-SPEKE* depende de la dificultad, como problemas, del intercambio Diffie-Hellman [12] [39] y de los logaritmos discretos [13] [29], [20] (pp. 3).

Como se hizo notar en el análisis del protocolo *AEKE* de la subsección 4.2.5, si un atacante obtiene k_1, de alguna manera, esto le permite realizar un ataque diccionario sobre P. Si k_1 es utilizada posteriormente en otra sesión, o permanece almacenada en la memoria del sistema por largo tiempo, la amenaza de robo es altamente probable. Esta amenaza también es aplicable a cualquier método extendido incluyendo este. Para prevenir esta amenaza, k_1 debe usarse sólo para la verificación de P y la derivación de k_2, pero inmediatamente después debe destruirse [20] (pp. 20).

El hecho de que A y B contribuyan a derivar la llave de sesión evita que un atacante que haya robado un verificador pueda convertirse en hombre enmedio, como sería el caso si la derivación de la llave fuera unilateral, y esto podría forzar el uso de la misma llave de A para B o viceversa. El atacante podría entonces suplantar a A enviando su propia función de prueba fp a B. El uso de Diffie-Hellman en el intercambio primario de la llave evita este ataque, ya que un atacante con el conocimiento de $h(P)$ no puede negociar la misma llave en dos sesiones distintas con A y B.

La función de prueba fp puede usar cualquier método conocido para probar conocimiento, como se hace en el protocolo *AEKE*. Es importante, no obstante, que la función de prueba combine cuidadosamente k_1 y k_2.

La llave k_2 no puede enviarse a B en claro o como resultado hash, debido a que k_2 se supone de baja entropía y por tanto el segundo intercambio Diffie-Hellman puede permitir un ataque diccionario.

También es importante que k_2 no sea enviada cifrada usando k_1 como llave, $E_{k_1}(k_2)$, porque entonces alguien que robe el verificador $h(P)$ puede realizar

[12] Sección A.11 del Apéndice A
[13] Subsección A.11.2 del Apéndice A

un ataque por hombre enmedio sobre el primer intercambio, descifrar k_2 de la función prueba de A, y cifrar de nuevo k_2 en una función de prueba para B. Además, si k_1 se cifra con k_2, $E_{k_2}(k_1)$, es posible un ataque diccionario por un atacante que pueda usar el conocimiento de todos los posibles valores de k_2 para determinar k_1.

De esta manera, la función prueba fp debe combinar k_1 y k_2, preservando las siguientes propiedades de ocultamiento de información:

1. El conocimiento de k_2 no debe revelar k_1
2. El conocimiento de k_1 no debe revelar k_2, con cálculos más rápidos que un ataque por búsqueda exhaustiva sobre k_2

Por otro lado, los problemas derivados de la baja entropía de P y, en consecuencia de k_2, se resuelven *protegiendo* el almacenamiento en B de V_1 y $h(P)$, y también por la prueba combinada de k_2 con la llave k_1 parcialmente autenticada.

4.6 Acerca de los Protocolos Fuertes

A partir de los protocolos presentados y analizados hasta este momento, parece paradójico que con pequeños passwords, criptográficamente débiles, se pueda lograr autenticación fuerte. Claramente el uso de passwords fuertes sería ideal, si las personas pudieran recordarlos. Tradicionalmente, la verificación de passwords sobre una red insegura, ha sido un problema difícil de resolver debido a la amenaza siempre presente de los ataques por diccionario. Los problemas alrededor de la autenticación basada en passwords habían durado tanto tiempo que se llegó a asumir que lograr autenticación fuerte usando sólo pequeños passwords era imposible. La presentación y análisis de los protocolos realizados en este capítulo han mostrado que el problema tiene soluciones viables en la medida en que se prueben su seguridad y eficiencia y se realicen implementaciones en diversos ámbitos donde se requieren esquemas basados en passwords. Probar la seguridad y eficiencia de protocolos para autenticación e intercambio de llaves es una tarea difícil, en particular para este tipo de protocolos basados en passwords. Esta dificultad tiene que ver con lo novedoso de las soluciones propuestas, con la

reciente aparición de estas direcciones de investigación, con la dependencia del análisis de seguridad respecto a los ataques para los que se asumen seguros los protocolos, con las escasas implementaciones que se tienen de estos protocolos, entre otros problemas.

A principios de los 90's, con la publicación de las primeras soluciones fuertes conocidas como protocolos *LGSN* [1, 6], debidas a Lomas, Gong, Saltzer, y Needham, un creciente interés sobre este problema fue produciendo resultados novedosos que aún se siguen estudiando hasta la fecha, como es el caso de la gran familia de protocolos *EKE* [7, 17] derivada de los trabajos originales de Bellovin y Merrit.

En este capítulo se han presentado y analizado los protocolos fuertes de la familia conocida como *LGSN* [1, 6], siendo las versiones para autenticación directa las más importantes desde el punto de vista del presente trabajo ya que se pretende lograr autenticación mutua y directa entre dos partes sin contar con una tercera parte confiable, tal como un servidor de llaves.

También se ha presentado y analizado el protocolo para autenticación mutua y directa entre dos partes conocido como *EKE* [7, 17]. De este protocolo se derivan un gran número de versiones tales como *EKE-DH* [17] (pp. 8) y [19] (pp. 4 - 5), y la extensión conocida como *AEKE* [7] (pp. 2 - 4), considerados como los más importantes por su influencia en posteriores extensiones. De estas extensiones, se ha presentado y analizado el protocolo *B-SPEKE*.

Las versiones y extensiones para autenticación mutua y directa entre dos partes de las familias *LGSN* y *EKE* se volverán a analizar en el siguiente capítulo de este trabajo desde un punto de vista comparativo, con el fin de resumir las propiedades y características que los diferencian entre sí.

También se ha presentado en este capítulo el protocolo conocido como *SPEKE* [19] el cual toma como base el protocolo *EKE-DH* para mejorar la resistencia a ataques por criptoanálisis respecto a las demás versiones de *EKE* [19].

Igualmente, se ha presentado el protocolo extendido *B-SPEKE* como ejemplo de uno de los métodos extendidos conocidos como *A-SPEKE*, *B-SPEKE*, y *B-EKE*, los cuales se presentan en [20] y que buscan mejorar la eficiencia de protocolos tales como *AEKE*.

La limitante de todos los métodos extendidos es que un verificador robado, por ejemplo $h(P)$, permite un ataque por fuerza bruta. La severidad de un ataque tal depende de la calidad de P, y también de la ingenuidad del ladrón. Conservadoramente se asume que un ataque por fuerza bruta sobre un verificador tal como $h(P)$, revelará P, y que se hace el "mejor esfuerzo" para mantener la base de datos con tales verificadores en secreto. También se hace notar que este requerimiento es común a todas las técnicas para autenticación mutua. El objetivo de estos métodos es pues, muy simple: proporcionar la mayor seguridad posible para P, ya sea este débil o fuerte criptográficamente hablando [20] (pp. 1).

Todos los protocolos presentados en este capítulo tienen en común la utilización de algún tipo de técnica criptográfica o combinación de ellas. Estas técnicas criptográficas varían entre criptografía de llave secreta, de llave pública, y funciones hash. Esta característica en común de los protocolos analizados está de acuerdo con la pretensión de este trabajo de estudiar protocolos fuertes para autenticación mutua y directa que utilicen técnicas criptográficas para lograr sus objetivos de seguridad.

Lo anterior viene al caso porque existen otros protocolos igualmente basados en passwords que no necesitan cifrar los mensajes intercambiados entre las partes para lograr sus objetivos de seguridad. Esta otra vertiente de protocolos utiliza técnicas de conocimiento cero (Sección A.8 del Apéndice A) donde las llaves públicas se intercambian en claro y pueden utilizarse en más de una ejecución del mismo o distinto protocolo. Aunque este tipo de protocolos no se analizó en este capítulo, en la Sección 5.1 del Capítulo 5 se mencionan protocolos de este tipo tales como *OKE* [41] y *SRP*-3 [11] como referencia al estado del arte en general de todas las soluciones basadas en passwords.

De la misma forma, en la Sección 5.2 del Capítulo 5 de este trabajo, también se hacen algunas comparaciones del protocolo *OKE* [41] con *EKE* y de *SRP*-3 [11] con *AEKE* y *B-SPEKE*, aunque pertenecen a vertientes de investigación distintas.

Capítulo 5

Análisis Comparativo de Protocolos Fuertes

La gran mayoría de protocolos fuertes basados en passwords que se han analizado en este trabajo tienen como objetivo principal evitar ataques diccionario fuera de línea. Pocos de los métodos que se han diseñado hasta la fecha cumplen con éxito este objetivo principal. Los protocolos analizados cumplen, o tratan de cumplir, de diferentes maneras sus objetivos de seguridad que, en algunos casos, no se concretan a evitar ataques diccionario sino que, además, presentan algunas otras características de seguridad adicionales.

Una pretensión de este trabajo es poder presentar un análisis, que de manera pormenorizada y también en forma resumida, muestre las características que cada uno de esos protocolos cumple, las suposiciones en que se basan, las técnicas que utilizan, los ataques que garantizan, sus ventajas, sus desventajas, y sus diferencias; todo esto de forma tal que se puedan comparar las características y propiedades de uno con las de los demás, de manera sencilla y rápida. Hasta la fecha, no se tiene conocimiento de que exista algún análisis comparativo similar entre protocolos de este tipo.

En este capítulo se hace un análisis del tipo antes mencionado, empezando por revisar los distintos protocolos, y sus referencias, que han intentado resolver el mismo problema que solucionan los protocolos fuertes, antes de que

estos surgieran y algunos otros posteriores. Después, se definen de manera precisa las características básicas que idealmente estos protocolos deben cumplir, y las características adicionales que sería deseable que cumplieran. Posteriormente, se revisan las principales características de los protocolos analizados en este trabajo, dando énfasis a las soluciones que ofrecen autenticación mutua y directa de dos partes, comparando, en lo posible, unos con otros. Como resultado de este análisis, se construye una tabla comparativa de todos los protocolos revisados, enfatizando las principales diferencias y características que los distinguen.

Finalmente, se construye una estructura arborescente del desarrollo y evolución en el tiempo que han tenido estos protocolos, mostrando sus principales influencias y derivaciones hasta el momento. Igualmente se menciona el estado de sus implementaciones hasta la fecha, su eficiencia, y las aplicaciones que este tipo de protocolos tiene actualmente y las que pudieran tener en el futuro.

5.1 Antecedentes y Actualidad de los Protocolos Fuertes

A lo largo de este trabajo se han explorado diversos enfoques de protocolos de autenticación fuerte basados en passwords, con distintas variaciones en sus grados de éxito y complejidad. No obstante, los protocolos presentados y analizados en este trabajo no son los únicos intentos que se han hecho por solucionar el problema de autenticación fuerte basada en passwords débiles. Antes y después de la aparición de los métodos fuertes, ha habido otros protocolos que han intentado resolver el mismo problema con distintos niveles de éxito.

En esta sección se hace un breve recorrido por las principales alternativas de solución que ha habido hasta el momento, mencionando algunos protocolos que sin ser considerados como fuertes, han intentado resolver el problema. Se presenta un resumen muy breve de sus características y, sobre todo, se mencionan las referencias donde se puede hacer una revisión más detallada de cada uno de estos protocolos. También se hace referencia a los protocolos fuertes presentados y analizados en el Capítulo 4, junto con algunos otros pertenecientes a vertientes de investigación no basadas en técnicas

criptográficas. Todo esto se hace con el fin de abarcar en esta sección la mayor cantidad posible de trabajos referentes al tema.

Un primer enfoque prometedor al problema, que se basa en criptografía de llave pública, fue el *Protocolo de Interbloqueo* (*"Interlock Protocol"*) [38] (pp. 49 - 55) de Shamir y Rivest. En este protocolo, A y B se envían mutuamente su llave pública en claro. Después, A cifra el mensaje que desea enviar a B con la llave pública de B, y le envía sólo la mitad de ese mensaje cifrado. Lo mismo hace B con A. Luego, A le envía la otra mitad de su mensaje cifrado a B, quien junta las dos mitades que recibió de A y lo descifra con su llave privada. También B envía su otra mitad de su mensaje a A, y este hace lo mismo que B para descifrarlo.

La idea de este protocolo es evitar ataques por hombre enmedio, comunes a los protocolos basados en criptografía de llave pública. Un ataque por hombre enmedio sobre este protocolo se describe en [56]. Un intento reciente para corregir esta debilidad se describe en [57].

El protocolo conocido como *estación a estación* descrito en [59] también trata de evitar ataques por hombre enmedio haciendo que A y B firmen sus respectivos mensajes que intercambian entre sí. Este protocolo asume que A tiene un certificado con la llave pública de B y que B tiene un certificado con la llave pública de A. Se supone que estos certificados han sido firmados por alguna autoridad confiable, ajena al protocolo. Al inicio del protocolo, A genera un número aleatorio x y lo envía a B, quien genera su propio número aleatorio y, usa Diffie-Hellman para calcular k a partir de x y y. B firma x y y cifrando la firma usando k como llave.

B envía esa firma cifrada, junto con y, a A. A calcula k de la misma forma que B, descifra la firma de B y la verifica. Entonces, A firma x y y enviando a B esa firma cifrada con k. Finalmente, B recibe la firma cifrada y verifica la firma de A.

Este protocolo es interesante desde el punto de vista comparativo puesto que depende del despliegue de llaves privadas y públicas. Utiliza un intercambio Diffie-Hellman y cifra los exponentes de manera similar a *EKE-DH* [17] (pp. 8) pero usando cifrado de llave pública.

En [1, 6] se describen los protocolos conocidos como *LGSN* de dos y tres partes basados en lo que se conoce como *llaves públicas secretas* y que se

presenta y analiza en la subsección 4.1.4 del capítulo 4 de este trabajo. Con estos protocolos, los autores pretenden hallar un método adecuado de llave pública en el que cualquier número aleatorio de longitud apropiada pueda ser una llave pública válida. En estos trabajos se describen protocolos que usan llaves públicas persistentes almacenadas, se discuten posibles ataques por diccionario, se analiza el *"login"* en Kerberos, y se presenta el importante concepto de *texto verificable* introducido por los autores.

En [36] se discuten algunos problemas concernientes a los protocolos *LGSN* [1, 6] y se sugieren modificaciones que evitan algunas de sus restricciones, tales como el uso de relojes sincronizados y la necesidad de mantener información de estado por parte del servidor. Como resultado de estos trabajos se proponen protocolos *LGSN* más simples y eficientes. En [55] se revisan los protocolos *LGSN* para optimizarlos en función del número de mensajes y rondas que utilizan. Esta revisión concluye con versiones optimizadas de estos protocolos.

Estos trabajos se utilizaron para presentar y analizar los protocolos de la familia *LGSN* de la sección 4.1 del capítulo 4 de esta tesis.

En [7, 17] se presentan y analizan los protocolos de la familia conocida como *EKE* que se han descrito en la sección 4.2 del capítulo 4 de esta tesis. En esos trabajos se presentan distintas versiones de los protocolos *EKE* basados en diferentes criptosistemas, resaltando la versión conocida como *EKE-DH* basada en intercambio de llave Diffie-Hellman. En [17] se analiza la seguridad de este protocolo en términos de fortalezas y debilidades, y se proponen mejoras y enmiendas.

En [18] se discuten posibles vulnerabilidades del protocolo básico *EKE* y su versión *EKE-DH* propuestos en [17]; se desarrollan mejoras y simplificaciones a esta última versión, dando como resultado un protocolo *EKE-DH* optimizado al que se le hace su análisis de seguridad.

Usando la teoría de números, en [60] se presentan ataques por diccionario sobre todas las versiones de los protocolos *EKE* discutidas en [17] y se ofrecen medidas para evitarlos. En este trabajo también se muestra de qué manera los confundidores aleatorios pueden no proteger contra ataques a las versiones para autenticación directa y llaves públicas secretas, ambas de los protocolos *LGSN* [1].

Basado en la versión *EKE-DH* [17], en [19] se presenta el protocolo conocido como *SPEKE* basado en intercambio exponencial de llave Diffie-Hellman, pero con la particularidad de que en lugar de la base para exponenciación comúnmente usada en *EKE-DH*, una función F transforma el password P en una base adecuada. En este trabajo se analizan los protocolos *EKE-DH* y *SPEKE* a la luz de ataques conocidos y otros nuevos, sugiriendo restricciones para prevenirlos. También se discuten beneficios, limitaciones, y conveniencias entre eficiencia y seguridad.

En [20] se presentan extensiones a las variantes de *EKE* conocidas como *EKE-DH* y *AEKE* [7], y de *SPEKE* [19], dando lugar a los métodos conocidos como *extendidos*. Estos métodos extendidos hacen de P un factor independiente en el proceso de autenticación, siendo los más conocidos los llamados *B-SPEKE*, *B-EKE*, y *A-SPEKE*, los cuales se presentan en el trabajo citado. También, en ese trabajo se resumen los problemas de implementación de esos métodos extendidos, y se muestra su potencial para mejorar su eficiencia con respecto al protocolo *AEKE*.

En [18] se muestra una versión minimizada de *EKE-DH* con tres mensajes, el cual se aplica igualmente bien a *SPEKE* [1].

Todos los anteriores trabajos referentes a los protocolos *EKE*, derivaciones y extensiones se han usado para presentar y analizar los protocolos de las secciones 4.2, 4.3, 4.4, y 4.5 del capítulo 4 de esta tesis. Se hace mención a ellos nuevamente porque la presente sección trata de dar un panorama resumido y rápido acerca de este tipo de protocolos y sus referencias principales.

En [65] se presenta el protocolo conocido como *negociación de llave fortificada* (*"Fortified Key Negotiation"*) cuyo objetivo principal es proteger el password de ataques diccionario y evitar ataques por hombre enmedio. Este protocolo utiliza funciones hash de dos variables que tienen una propiedad especial: tienen muchas colisiones sobre la primera variable y ninguna colisión sobre la segunda variable. Que una función hash sea libre de colisiones o resistente a colisiones significa que es computacionalmente infactible hallar dos distintas entradas x, y las cuales mapeen al mismo valor hash, es

[1] El que un protocolo con un número mínimo de mensajes sea verdaderamente óptimo depende del objetivo deseado y del costo y velocidad de la comunicación, comparado con el costo computacional. Por ejemplo, optimizar *SPEKE* para tiempo mínimo de ejecución puede requerir más de tres mensajes para maximizar el procesamiento paralelo de ambas partes [19]

decir, $h(x) = h(y)$. Una función hash tal puede expresarse como: $H'(k,x) = H(H(k,x) \ mod \ 2^m, x)$, donde $H(k,x)$ es una función hash ordinaria sobre k y x.

El protocolo inicia suponiendo que A y B comparten P y que han acordado k usando Diffie-Hellamn. Primero, A envía a B, $H'(P,k)$; luego, B calcula $H'(P,k)$ y compara su resultado con el que recibió de A. Si coinciden, B envía a A, $H'(H(P,k))$. Finalmente, A calcula $H'(H(P,k))$ y compara su resultado con el recibido de B.

El punto en este protocolo consiste en que con este tipo de función hash, muchos passwords son igualmente probables de producir el mismo valor cuando se calcula la función con P y una cierta k_1. Un probable atacante por hombre enmedio tendría que compartir k_1 con A o con B. Esto significa que cuando el posible atacante pudiera hallar una coincidencia, P puede ya no ser válido pues A o B ya se habrían dado cuenta del intento de ataque por los rechazos registrados, y podrían haber cambiado P.

Otras variaciones sobre el tema son los protocolos de tres partes donde se comparten múltiples secretos y se guardan en un servidor de autenticación común y confiable, el cual interviene como mediador en el proceso de intercambio entre dos partes. En [18] se muestra una debilidad en algunas instancias de esos protocolos.

Aún lejos del espectro de los métodos basados en passwords están los esquemas basados en identidad [38] (pp. 115), los cuales pueden usar ideas de conocimiento cero para probar el conocimiento de un secreto que otra parte conoce. Los esquemas basados en conocimiento cero no proporcionan intercambio de llave autenticada y no permiten que el secreto sea criptográficamente pequeño. Una breve revisión de estos métodos se puede encontrar en [59].

En [41] se presenta un novedoso protocolo conocido como OKE (*"Open Key Exchange"*) que se basa en sistemas de llave pública pero con la novedad que las llaves públicas se intercambian en claro, sin cifrarlas. Además, tiene la ventaja que la misma pareja de llaves pública/privada se puede utilizar en tantas ejecuciones del protocolo como se desee. En ese trabajo se muestra lo eficiente y práctica que resulta la variante de este protocolo basada en ***RSA***, así como también una comparación interesante entre este protocolo y ***EKE***, la cual se resume en la siguiente sección de esta tesis.

En [11] se presenta un nuevo protocolo genéricamente conocido como *SRP* (*"Secure Remote Password Protocol"*) que resiste ataques diccionario, ofrece seguridad hacia adelante, y permite que los passwords se almacenen de tal forma que no presenten *equivalencia a texto en claro* al password mismo, evitando con esto que un atacante que obtenga el archivo de passwords pueda comprometer la seguridad, y tener acceso inmediato al servidor. En ese mismo trabajo se analiza *SRP* y se propone una versión mejorada conocida como *SRP-3*. Este protocolo combina técnicas de conocimiento cero con protocolos de intercambio de llave asimétricos, ofreciendo una eficiencia significativamente mejor con respecto a métodos extendidos comparables, como *AEKE* y *B-SPEKE* [20].

En [58] se presentan tres protocolos basados en *RSA*, Diffie-Hellman, y ElGamal respectivamente. Estos protocolos pretenden ser más simples y rápidos que sus predecesores, garantizando una seguridad ligeramente mayor; en particular, no cifran la llave de sesión k, y por tanto no imponen ninguna restricción a su uso posterior. Suponiendo que las partes comparten P y son capaces de generar números aleatorios, estos protocolos inician intercambiando un secreto criptográficamente fuerte pero no compartido y terminan compartiendo una llave de sesión fuerte y compartida. A estos protocolos basados en llave pública los autores los llaman genéricamente *S3P* (*"Strong Secret Sharing Password Protocol"*).

En [61] se estudian protocolos en escenarios asimétricos donde el servidor de autenticación posee una pareja de llaves pública/privada, mientras que el cliente tiene sólo un password P como llave de autenticación. El análisis muestra que esos protocolos son óptimamente resistentes a ataques diccionario. Además, para autenticación de usuarios, se presentan protocolos que proporcionan autenticación de dos partes, intercambio de llave autenticada, defensa contra compromisos del servidor, y anonimato del usuario. También se realiza una prueba que muestra que las técnicas de llave pública son inevitables para protocolos basados en passwords que pretendan resistir ataques diccionario. Se introduce asímismo, la noción de *passwords públicos* que se usan en tales protocolos para situaciones donde la máquina del cliente no cuenta con los recursos de cómputo para validar la llave pública del servidor. Estos passwords públicos se pueden conceptualizar como compendios de la llave pública del servidor; el usuario no necesita memorizarlos, puede escribirlos o almacenarlos en algún lado, y puede usarlos sin necesidad de dispositivos especiales de cómputo.

5.2 Análisis Comparativo

En esta sección se hace un análisis comparativo de los protocolos fuertes basados en passwords, haciendo énfasis en los protocolos que ofrecen autenticación mutua y directa entre dos partes. Este análisis se hace en función de las características que cada uno de esos protocolos cumple, las suposiciones en que se fundamentan, las técnicas que utilizan, los ataques que resisten, sus ventajas, desventajas, y diferencias. El análisis trata, hasta donde es posible, de comparar las características y propiedades de uno con las de los demás, de manera sencilla y rápida. Finalmente, se construye una tabla comparativa que resume las características principales de cada método, y una estructura arborescente que muestra la evolución de estos métodos en el tiempo, sus influencias y derivaciones.

Para llevar a cabo todo lo anterior, se empiezan por definir las características fundamentales o básicas de estos métodos. Después, también se definen las características que serían deseables que estos métodos cumplieran, además de las básicas.

5.2.1 Características de los Protocolos Fuertes

Para que un protocolo de autenticación sea considerado como fuerte, se requiere que cumpla con algunas características que lo distingan de otros protocolos de autenticación. Hay un grupo de estas características que son básicas y que deben cumplir todos los métodos fuertes; existen algunas otras que sería deseable que cumplieran [19] (pp. 3).

Debido a que la mayoría de estas características ya fueron analizadas en las secciones 2.3 del Capítulo 2 y 3.4 del Capítulo 3, se listarán sin detallarlas. Sólo se describirán aquellas que no hayan sido mencionadas en capítulos anteriores.

Las características consideradas como básicas o fundamentales [19] (pp. 3), son las siguientes:

1. ***Impedir ataques diccionario fuera de línea***

Este es el ataque que la mayoría de los protocolos de este tipo trata de evitar y se describe con detalle en la subsección 2.3.3 del capítulo 2. Cuando se menciona genéricamente un ataque por diccionario, se asume que se hace referencia a un ataque fuera de línea.

2. *Resistir ataques diccionario en línea*

   Estos ataques y la forma de prevenirlos se tratan en la subsección 2.3.3 del capítulo 2. Debido a que un ataque diccionario en línea es fácilmente detectable y se puede evitar de manera sencilla, no es considerado en el análisis comparativo de protocolos fuertes.

3. *Proporcionar autenticación mutua*

   Este concepto se describe en la sección 1.2 del capítulo 1 de este trabajo y tiene que ver con el hecho de que todos los participantes en el protocolo deben identificarse entre sí; es decir, mutuamente.

4. *Integrar intercambio de llave*

   En la subsección 1.2.1 del capítulo 1, se describe el concepto de intercambio de llave, y en la sección 3.4 del capítulo 3 se hace lo mismo con los protocolos de intercambio de llaves. En todo este trabajo se ha asumido que el intercambio de llaves es un proceso integrado al de autenticación.

5. *Que el usuario no necesite registrar y almacenar secretos persistentes*

   El que sea innecesario registrar y almacenar secretos persistentes significa que el usuario no requiere guardar llaves secretas, públicas, o privadas. Existen varias maneras de crear un canal seguro para enviar un password en claro o para intercambiar un resultado hash de él, pero el objetivo de estos protocolos es, sin embargo, más ambicioso: hacer del password un factor independiente en el proceso de autenticación. De no ser así, los secretos persistentes tendrían que ser generados, distribuidos, y almacenados de modo seguro, lo que presenta problemas adicionales. Esos secretos no deben revelarse nunca y no deben exponerse a robo o falsificación, ya que esto agregaría debilidades adicionales para vulnerar esos protocolos.

   Se pueden construir sistemas donde la seguridad del password dependa de una llave almacenada, pero de esta manera se traslada la base de seguridad, del password a la llave. Si la llave es robada, el password se compromete. El eliminar llaves persistentes elimina este problema y evita la necesidad de almacenamiento seguro.

Las características anteriores se consideran fundamentales para los protocolos de autenticación basados en passwords. En [59], se discute la seguridad general de los protocolos de intercambio de llave autenticada (Sección 3.4 del Capítulo 3) y se describen sus características. Estas características para protocolos de intercambio de llaves se agregan a las anteriormente descritas para protocolos de autenticación, ya que en este trabajo se asume la autenticación como un proceso que también incluye el intercambio de llaves.

Las características deseables [19] (pp. 13) que se agregan, son las siguientes:

1. *Seguridad Perfecta hacia Adelante*

   Esta propiedad se describe con detalle en la Sección 3.4 del Capítulo 3 de este trabajo y se refiere al no compromiso de llaves de sesión pasadas a partir de la eventualidad de comprometer llaves permanentes, ya sean estas secretas o públicas.

2. *Autenticación Directa*

   Esta característica se describe en la Sección 1.2 del Capítulo 2 de este trabajo y se refiere a la autenticación que se realiza solo con la participación de las partes directamente involucradas en el proceso, sin intervención de nadie más. El análisis de los protocolos fuertes se hace, de hecho, sobre versiones que ofrecen autenticación mutua y directa, por lo que no será necesario mencionarla explícitamente.

3. *No Instantáneas*

   El concepto de instantánea se describe en la subsección 2.2.2 del Capítulo 2 de este trabajo. Este requerimiento significa que los mensajes no deben contener datos dependientes del tiempo, los cuales requieren sincronización de relojes entre las partes.

4. *Ocultamiento de Identidad*

   Esta característica es difícil de cumplir. Se trata de ocultar la identidad del usuario a observadores pasivos. Ninguno de los protocolos descritos en este trabajo satisface esta característica. Esta limitación puede ser superada con un intercambio Diffie-Hellman adicional y previo a la ejecución del protocolo para establecer una llave autenticada que sirva para cifrar la identidad del usuario. Un ataque por hombre enmedio podría revelar esta identidad oculta, tal vez a costa de ser detectado. El protocolo estación a estación [38] (pp. 516) oculta la identidad de

los participantes a un atacante pasivo; no obstante, también podría revelar la identidad a un atacante por hombre enmedio, pero al precio de romper la ejecución del protocolo [19] (pp. 13).

El problema de autenticación con ocultamiento de identidad parece ser intratable si no se utiliza una sesión segura previamente establecida, o grandes llaves públicas. En estos métodos, el problema se relaciona al hecho de que el secreto es compartido. Para elegir el secreto apropiado para una de las partes, se necesita conocer la identidad de esa parte antes del intercambio. No obstante, antes del intercambio se asume que no existe un canal seguro para enviar la identidad [19] (pp. 13).

En las siguientes subsecciones se analizan los principales protocolos presentados en el capítulo 4 de este trabajo, tomando como base las principales referencias conocidas que se mencionan en la sección 5.1 de este capítulo, y complementando la comparación con observaciones propias cuando es necesario.

5.2.2 LGSN y EKE

Las familias de protocolos *LGSN* y *EKE* son las dos ramas más importantes de protocolos fuertes analizadas en este trabajo. Las versiones de estas familias para autenticación mutua y directa se presentan y analizan en las subsecciones 4.1.5 y 4.2.1, respectivamente, del capítulo 4 de este trabajo.

LGSN para Autenticación Directa

La versión de *LGSN* para Autenticación Directa, presentado en la subsección 4.1.5 del capítulo 4 de esta tesis, tiene las siguientes propieades y características, que se resumen en la tabla presentada en la Figura 1 de la subsección 5.2.7 de este capítulo:

1. **Servicios que proporciona**
   (a) *Autenticación mutua*
   (b) *Intercambio integrado de llave pública y de llave de sesión*

2. **Suposiciones en las que se basa**

   (a) *A y B comparten el conocimiento sobre P*

3. **Ataques que Soporta**

   (a) *Diccionario*

   (b) *Texto Verificable*

   (c) *Réplica*

4. **Particularidades**

   (a) *Usa P para cifrar la llave pública de A*

   (b) *En el mensaje 2,* $(E_{K_A^p}(B,\ A,\ N_{B_1},\ N_{B_2},\ C_B,\ E_P(Ch_A)))$*, se realiza un doble cifrado*

   (c) *Usa una función hash del quienvive para autenticación y saludo*

   (d) *Utiliza núnicos*

5. **Ventajas**

   (a) *No utiliza instantáneas*

   (b) *No requiere intercambio previo de llave pública*

6. **Desventajas**

   (a) *La llave de sesión debe viajar por el canal*

   (b) *La llave privada de A, en poder de A, debe mantenerse en secreto*

   (c) *La llave pública de A, conocida por A y B, no debe revelarse*

   (d) *A debe generar las llaves pública, privada, y de sesión para cada ejecución*

Dadas las anteriores características, la versión *LGSN* para Autenticación Directa cumple con los 4 primeros requerimientos básicos para este tipo de protocolos, a saber: impide ataques diccionario tanto en línea como fuera de línea, proporciona autenticación mutua y directa, e integra el intercambio de llave. El requerimiento de evitar el almacenamiento de secretos persistentes no se cumple debido a que el protocolo necesita que *A* almacene su llave privada. Además, el protocolo cumple con los requerimientos adicionales de autenticación directa y el no uso de instantáneas.

LGSN Reforzado para Autenticación Directa

En [1] (pp. 13) se propone una modificación al protocolo anterior que supera la debilidad de que la llave de sesión k es unilateralmente generada por A y, sobre todo, dado que esa llave viaja por el canal de comunicaciones, evita ataques por criptoanálisis sobre k. Esta modificación, referenciada como *LGSN* Reforzado para Autenticación Directa, se describe en la Sección 4.1.6 del Capítulo 4 de este trabajo.

Las características y propiedades de esta nueva versión se muestran a continuación y se resumen en la tabla que se ilustra en la Figura 1 de la subsección 5.2.7 de este capítulo:

1. **Servicios que proporciona**
   (a) *Autenticación mutua*
   (b) *Intercambio integrado de llave pública*
   (c) *Establecimiento de llave de sesión*
2. **Suposiciones en las que se basa**
   (a) *A conoce P*
   (b) *B* conoce $h(P)$
3. **Ataques que Soporta**
   (a) *Diccionario*
   (b) *Texto Verificable*
   (c) *Réplica*
   (d) *Criptoanálisis sobre k*
4. **Particularidades**
   (a) *Usa $h(P)$ para cifrar la llave pública de A*
   (b) *En el mensaje 2, $(E_{K_A^p}(B, A, N_{B_1}, N_{B_2}, C_B, Ch_B, E_{h(P)}(Ch_A)))$, se realiza un doble cifrado*
   (c) *Usa una función hash del quienvive para autenticación y saludo*
   (d) *Utiliza núnicos*

5. **Ventajas**

   (a) *No utiliza instantáneas*

   (b) *No requiere intercambio previo de la llave pública*

   (c) *Elección bilateral de k*

   (d) *La llave de sesión k no se intercambia. Ambas partes contribuyen para calcularla*

6. **Desventajas**

   (a) *La llave privada de A, en poder de A, debe mantenerse en secreto*

   (b) *La llave pública de A, conocida por A y B, no debe revelarse*

   (c) *A debe generar la llaves pública, privada, y de sesión para cada ejecución*

Dadas las anteriores características, esta versión de *LGSN* Reforzado para Autenticación Directa también cumple con los 4 primeros requerimientos básicos para este tipo de protocolos, a saber: impide ataques diccionario tanto en línea como fuera de línea, proporciona autenticación mutua, e integra el establecimiento de llave. El requerimiento de evitar el almacenamiento de secretos persistentes no se cumple debido a que el protocolo necesita que A almacene su llave privada. Además, este protocolo cumple con los requerimientos adicionales de autenticación directa y el no uso de instantáneas.

La ganancia de esta versión respecto a la versión original consiste en que utiliza $h(P)$ en lugar de P para cifrar los mensajes 1 $(Ch_A,\ E_P(K_A^p))$, y 3 $(E_P(N_{B_1},\ k \oplus N_{B_2}))$, lo cual complica aún más un posible ataque por diccionario. También, y sobre todo, protege a k de la posibilidad de ataques por criptoanálisis al permitir que la generación de k sea unilateral, y esto evita la necesidad de intercambiarla a través del canal de comunicaciones.

EKE Original

El protocolo *EKE* original, que se describe en la subsección 4.2.1 del Capítulo 4 de este trabajo, fue diseñado para proporcionar autenticación directa y mutua de dos partes. Este protocolo tiene las siguientes características y propiedades, que también se resumen en la tabla mostrada en la Figura 1 de la subsección 5.2.7 de este capítulo.

1. **Servicios que proporciona**
   (a) *Autenticación mutua*
   (b) *Intercambio integrado de llave pública y de llave de sesión*
2. **Suposiciones en las que se basa**
   (a) *A y B comparten el conocimiento sobre P*
3. **Ataques que Soporta**
   (a) *Diccionario*
   (b) *Texto Verificable*
   (c) *Réplica*
4. **Particularidades**
   (a) *Usa P para cifrar la llave pública de A*
   (b) *El mensaje 2, $(E_P(E_{K_A^p}(k)))$, se cifra doblemente*
   (c) *Usa quienvives para autenticación y saludo*
   (d) *No usa núnicos*
5. **Ventajas**
   (a) *No utiliza instantáneas*
   (b) *No requiere intercambio previo de llave pública*
   (c) *El servidor B genera la llave de sesión*
6. **Desventajas**
   (a) *La llave de sesión debe viajar por el canal*
   (b) *La llave privada de A, en poder de A, debe mantenerse en secreto*
   (c) *La llave pública de A, conocida por A y B, no debe revelarse*
   (d) *A debe generar la llaves pública y privada en cada ejecución*

De acuerdo a estas características, el protocolo *EKE* Original cumple los cuatro primeros requerimientos básicos para protocolos de este tipo, a saber: impide ataques diccionario tanto en línea como fuera de línea, proporciona autenticación mutua, e integra el establecimiento de llave. El requerimiento de evitar el almacenamiento de secretos persistentes no se cumple debido a

que A debe mantener las llaves pública y privada en secreto durante toda la ejecución del protocolo. Por lo que respecta a los requerimientos adicionales deseables en este tipo de métodos, este protocolo cumple con los de autenticación directa y no uso de instantáneas.

EKE Reforzado

Considerando la eventual posibilidad de un ataque usando criptoanálisis, el cual otorgaría al atacante conocimiento sobre la llave k, dándole de esta manera una vía para atacar P por diccionario, en [17] se propone una variación menor al protocolo para prevenir la posibilidad de este ataque.

Esta variación al protocolo EKE consiste en que durante el intercambio quienvive-respuesta, A y B generen subllaves aleatorias k_A y k_B, de tal modo que sean estas subllaves las que se intercambien y a partir de ellas cada parte calcule la verdadera llave de sesión $k = f(k_A, k_B)$ sin necesidad de intercambiarla. No obstante la anterior protección, puede concebirse un ataque criptoanalítico más sofisticado que explote la presencia de quienvives y respuestas en diferentes mensajes para atacar k. Esto parece improbable; no obstante, por si fuera de interés, en [17] se propone una mejora consistente en modificar las respuestas para que contengan una función hash unidireccional g de los quienvives, y no los quienvives mismos.

El protocolo EKE Reforzado se describe en la Subsección 4.2.2 del Capítulo 4, y tiene las siguientes características y propiedades que se resumen en la tabla mostrada en la Figura 1 de la subsección 5.2.7 de este capítulo.

1. **Servicios que proporciona**
   (a) *Autenticación mutua*
   (b) *Intercambio integrado de llave pública*
   (c) *Establecimiento de llave de sesión*
2. **Suposiciones en las que se basa**
   (a) *A y B conocen P*
3. **Ataques que Soporta**

(a) *Diccionario*

(b) *Texto Verificable*

(c) *Réplica*

(d) *Criptoanálisis sobre k*

4. **Particularidades**

(a) *Usa P para cifrar la llave pública de A*

(b) *El mensaje 2, $(E_P(E_{K_A^p}(k_i)))$, se cifra doblemente*

(c) *Usa una función hash del quienvive para autenticación y saludo*

(d) *No usa núnicos*

5. **Ventajas**

(a) *No utiliza instantáneas*

(b) *No requiere intercambio previo de llave pública*

(c) *Elección bilateral de k*

(d) *La llave de sesión k no se intercambia*

6. **Desventajas**

(a) *La llave privada de A, en poder de A, debe mantenerse en secreto*

(b) *La llave pública de A, conocida por A y B, no debe revelarse*

(c) *A debe generar la llaves pública, privada, y de sesión para cada ejecución*

(d) *Relativa complejidad*

Dadas las anteriores características, esta versión Reforzada de EKE también cumple con los 4 primeros requerimientos básicos para este tipo de protocolos: impide ataques diccionario tanto en línea como fuera de línea, proporciona autenticación mutua, e integra el establecimiento de llave. El requerimiento de evitar el almacenamiento de secretos persistentes no se cumple debido a que A debe mantener las llaves pública y privada en secreto durante toda la ejecución del protocolo. Por lo que respecta a los requerimientos adicionales deseables en este tipo de métodos, este protocolo cumple con los de autenticación directa y no uso de instantáneas.

La ganancia de esta versión respecto a la versión original consiste en que protege a k de la posibilidad de ataques por criptoanálisis al permitir que la generación de k sea unilateral, y esto evita la necesidad de intercambiar a a través del canal de communicaciones. Esto a cambio de hacer un poco más complejo el protocolo.

Comparación de LGSN con EKE

Comparando las versiones originales de *LGSN* y *EKE*, ambos protocolos proporcionan los mismos servicios, usando intercambios ligeramente distintos y presentando debilidades similares. Con el fin de corregir estas debilidades comunes, se propusieron versiones reforzadas de cada uno de ellos que corrigen dos deficiencias fundamentales: reforzar la seguridad de los quienvives a través de una función hash, y evitar ataques por criptoanálisis sobre k haciendo que las partes la calculen bilateralmente, evitando así que k viaje por el canal de comunicaciones.

Algunos intentos de comparar estos protocolos se han realizado utilizando distintos parámetros de comparación, como son la eficiencia en el número de mensajes y de rondas, entre otros. A continuación, se describen algunos de estos intentos comparativos y sus referencias.

En [55], Gong analiza los protocolos *LGSN*, *EKE*, y variantes descritos en [1, 6, 7, 17] a partir del hecho de que estos protocolos, que brindan protección contra ataques por diccionario, son más costosos en términos de número de mensajes y rondas que aquellos protocolos de autenticación que no tienen este requerimiento de seguridad [10].

Hasta entonces se creía que tales incrementos en los costos en mensajes y rondas eran inherentes a la naturaleza de tales protocolos y, en particular, debidos a que el servidor debe decidir si el requerimiento de un cliente es fresco, antes de dar una respuesta. En [55] se muestra que esta restricción es incidental a las técnicas usadas en esos protocolos, y que utilizando un diseño diferente se pueden desarrollar protocolos de autenticación resistentes a ataques por diccionario y, al mismo tiempo, óptimos en número de mensajes y rondas [55] (pp. 1).

En [55] (pp. 4) se presentan dos versiones de *LGSN* con Autenticación

Directa de dos partes optimizadas a tres mensajes y tres rondas, que es la cota inferior probada [10] (pp. 10 - 12) para protocolos basados en núnicos con saludo. Una de esas versiones involucra la generación unilateral de k por parte de B y en la otra versión participan A y B en la generación de k. Este versión optimizada de *LGSN* se presenta en la subsección 4.1.7 del capítulo 4 de esta tesis.

En [36], Tsudik y Van Herreweghen sugieren modificaciones al protocolo *LGSN* con el fin de reducir la cantidad de cifrado. Sus técnicas incluyen el uso de llaves de cifrado generadas por el usuario y la reducción en el número de mensajes [55] (pp. 4). No obstante, en estas modificaciones sugeridas, *A no sabe si la llave de sesión* k que recibe de B es correcta sino hasta que la usa más tarde. Esta deficiencia, que parece ser intrínseca, es sin embargo compensada por el uso de mensajes extremadamente cortos en los protocolos que se proponen en [55] (pp. 4 - 5).

En [1] (pp. 15) Gong y sus coautores consideran que su versión del protocolo *LGSN* con Autenticación Directa (Subsección 4.1.5 del capítulo 4) tiene clara ventaja sobre el protocolo *EKE* (Subsección 4.2.1 del capítulo 4) debido a que el uso de confundidores y núnicos protegen de manera tan segura la llave de sesión k y el password P, que el impacto de comprometerla (debido, por ejemplo, a una gran cantidad de tráfico o a criptoanálisis exitoso) está estrictamente limitado a los mensajes de esa sesión.

El argumento para la afirmación anterior es que, en el protocolo *EKE*, comprometer la llave de sesión k podría permitir a un atacante replicar el mensaje 2 de *EKE* ($B \longrightarrow A : E_P(E_{K_A^p}(k))$, y hacerse pasar por B en todas las sesiones futuras. Además, se argumenta, *EKE* permitiría ataques por diccionario porque se puede conjeturar P, descifrar el mensaje 1 de *EKE* ($A \longrightarrow B$: A, $E_P(K_A^p)$) para obtener k, y usar (P, K_A^p, k) para reconstruir el mensaje 2 y verificar la conjetura.

En [60] se presentan ataques por diccionario usando resultados de la teoría de números sobre todas las versiones de *EKE* discutidas en [17]. También se muestra de qué manera los confundidores aleatorios pueden no brindar protección a las versiones de Autenticación Directa y Llaves Públicas Secretas de la familia *LGSN* contra ataques diccionario.

Hasta aquí, se han podido comparar las versiones originales y reforzadas de los protocolos *LGSN* y *EKE*. Los subsecuentes trabajos de investigación en

este tipo de protocolos han explorado más las extensiones y posibilidades de *EKE*, a partir de las cuales han surgido un buen número de ideas que se han concretado en nuevas versiones y han dado origen a lo que se conoce como métodos extendidos [7, 19, 20]. Respecto a la familia de protocolos *LGSN*, se han presentado optimizaciones [18] a partir de la versión reforzada, y su influencia actual está presente en los protocolos Halevi-Crawczyk [61] y *S3P* [58], ambos basados en sistemas de llave pública. Esta influencia se muestra esquemáticamente en la estructura arborescente de la Figura 2 en la sección 5.3 de este capítulo.

5.2.3 EKE-DH y AEKE

Los protocolos *EKE-DH* y *AEKE*, presentados en la subsección 4.2.3 y 4.2.5, respectivamente, del capítulo 4 de esta tesis son dos de las versiones más interesantes de la familia de protocolos *EKE*. Por esta razón se resumen en esta subsección las características y propiedades más representativas de cada uno de ellos.

EKE-DH

A partir del hecho de que el protocolo *EKE* utiliza cifrado asimétrico para intercambiar una llave de sesión que se usa en un sistema de cifrado simétrico, surge la versión de *EKE* con intercambio Diffie-Hellman, conocida como *EKE-DH*, y descrita en la Sección 4.2.3 del Capítulo 4 de este trabajo.

Las características y propiedades de *EKE-DH*, que se resumen en la tabla presentada en la Figura 1 de la Subsección 5.2.7 de este Capítulo, son las siguientes:

1. **Servicios que proporciona**
   (a) ***Autenticación mutua***
   (b) ***Establecimiento de llave de sesión***
   (c) ***Seguridad hacia adelante***

2. **Suposiciones en las que se basa**

   (a) *A y B conocen P*

   (b) *A y B acuerdan la base α y el módulo β*

3. **Ataques que Soporta**

   (a) *Diccionario*

   (b) *Texto Verificable*

   (c) *Réplica*

   (d) *Criptoanálisis sobre k*

4. **Particularidades**

   (a) *Usa intercambio Diffie-Hellman*

   (b) *Usa P para cifrar $\alpha^{R_A} \pmod{\beta}$ y $\alpha^{R_B} \pmod{\beta}$*

   (c) *Usa quienvives para autenticación y saludo*

   (d) *No usa núnicos*

5. **Ventajas**

   (a) *No utiliza instantáneas*

   (b) *No requiere intercambio previo de llave pública*

   (c) *Elección bilateral de k*

   (d) *La llave de sesión k no se intercambia*

6. **Desventajas**

   (a) *Relativa dificultad del acuerdo previo de α y β, porque este acuerdo es ajeno al protocolo*

De acuerdo a estas características, *EKE-DH* cumple con todos los requerimientos básicos para este tipo de protocolos: impide ataques diccionario dentro y fuera de línea, proporciona autenticación mutua, integra el acuerdo de llave de sesión, y cumple con el requisito de que la parte A no almacene secretos persistentes. Además, cumple con los requerimientos adicionales de proporcionar seguridad perfecta hacia adelante, autenticación mutua, y no uso de instantáneas.

La ganancia espectacular, por su sencillez, de esta versión respecto a las versiones original y reforzada, tanto de *EKE* como de *LGSN*, consiste en que a los requerimientos básicos agrega el no necesitar almacenar secretos persistentes, ya que a pesar de que A debe guardar $\alpha^{R_A} \ (mod\ \beta)$, este no es un secreto, y su conocimiento por parte de un atacante no compromete la seguridad del protocolo.

Otra ventaja importante consiste en que ofrece seguridad hacia adelante, debido a que el eventual compromiso de una llave de sesión no compromete ni la información pasada ni futura del protocolo en cuanto a secretos se refiere. Esto se debe a que los valores exponenciales que se intercambian dependen de números aleatorios generados para cada ejecución del protocolo, y porque, además, esos valores no revelan ninguna información, ni sobre la llave de sesión k, ni sobre P.

AEKE

A pesar de las mejoras fundamentales de la versión *EKE-DH* sobre todas las anteriores, requiere que ambas partes, A y B, tengan versiones en claro de P, una restricción que no siempre puede cumplirse.

En la Sección 4.2.5 del Capítulo 4 de este trabajo se describe una extensión de *EKE-DH*, conocida como *AEKE* dentro de los métodos extendidos, para manejar la situación cuando del lado de B se guarda el resultado de aplicar una función hash $h(P)$ a P, y la validación de P se realiza calculando $h(P')$ del P' tecleado, y comparándolo con el valor almacenado.

En esta versión, A y B usan $h(P)$, el cual debe ser secreto. No obstante, dado que se asume que bajo alguna circunstancia $h(P)$ puede comprometerse, A debe enviar un mensaje adicional conteniendo una diferente función hash de P; este valor, junto con $h(P)$ y la llave de sesión, es usado por B para validar la secuencia de *"login"*.

A los objetivos de seguridad de *EKE-DH*, esta versión agrega la protección contra compromisos del archivo de passwords; es decir, protege contra la posibilidad de que el archivo que contiene los resultados hash de P sea robado. Dos maneras de lograr estos objetivos son: una por medio del uso

de firmas digitales [2], y la otra basándose en una familia de funciones hash unidireccionales conmutativas [7] (pp. 4).

Las características y propiedades de *AEKE*, que se resumen en la tabla mostrada en la Figura 1 de la Subsección 5.2.7 de este Capítulo, son las siguientes:

1. **Servicios que proporciona**
   (a) *Autenticación mutua*
   (b) *Establecimiento de llave de sesión*
   (c) *Seguridad hacia adelante*
2. **Suposiciones en las que se basa**
   (a) *A conoce P*
   (b) *B conoce h(P)*
   (c) *A y B acuerdan la base α y el módulo β*
3. **Ataques que Soporta**
   (a) *Diccionario*
   (b) *Texto Verificable*
   (c) *Réplica*
   (d) *Criptoanálisis sobre k*
   (e) *Compromiso de archivo conteniendo h(P)*
4. **Particularidades**
   (a) *Usa intercambio Diffie-Hellman*
   (b) *Usa h(P) para cifrar $\alpha^{R_A} \pmod{\beta}$ y $\alpha^{R_B} \pmod{\beta}$*
   (c) *Usa quienvives para autenticación y saludo*
   (d) *Usa f(P, k) para proteger compromisos de h(P)*
   (e) *No usa núnicos*
5. **Ventajas**

[2] Sección A.5 del Apéndice A

(a) *No utiliza instantáneas*

(b) *No requiere intercambio previo de llave pública*

(c) *Elección bilateral de k*

(d) *La llave de sesión k no se intercambia*

(e) *Resiste compromisos de archivo conteniendo $h(P)$*

(f) *No es posible suplantar a A ante B*

6. Desventajas

(a) *Relativa dificultad del acuerdo previo de α y β, ya que es un acuerdo ajeno al protocolo*

(b) *Posible suplantar a B ante A*

(c) *Agrega un intercambio más, $E_k(f(P,k))$, con respecto al protocolo $EKE\text{-}DH$*

De acuerdo a estas características, este protocolo cumple con todos los requerimientos básicos para este tipo de protocolos, a saber: impide ataques diccionario dentro y fuera de línea, proporciona autenticación mutua, integra el acuerdo de llave de sesión, y cumple con el requisito de que la parte A no almacene secretos persistentes. Además, cumple con los requerimientos adicionales de proporcionar seguridad perfecta hacia adelante, autenticación mutua, y no usa instantáneas.

La ganancia de esta versión respecto a la versión $EKE\text{-}DH$ consiste en que relaja la restricción acerca del conocimiento que ambas partes deben tener acerca de P, y en que agrega la protección adicional para el caso en que se comprometa el archivo que contiene el resultado hash de P. En esta versión, A conoce P; B puede conocer sólo el resultado hash de P.

5.2.4 EKE-DH y SPEKE

Por su importancia dentro de los métodos conocidos como extendidos, en esta subsección se presenta un resumen de las características y propiedades del protocolo $SPEKE$, presentado y analizado en la subsección 4.3.1 del capítulo 4 de esta tesis. También se presenta en esta subsección una comparación de $SPEKE$ con el protocolo $EKE\text{-}DH$, en el cual se basa.

SPEKE

Basado fundamentalmente en *EKE-DH*, el método extendido conocido como *SPEKE* descrito en la subsección 4.3.1 del Capítulo 4 de este trabajo, busca reforzar la seguridad de *EKE-DH* y *AEKE* contra ataques criptoanalíticos. Para lograrlo, su principal innovación consiste en convertir a P, por medio de una función F, en una base adecuada para Diffie-Hellman, evitando de esta manera tener que cifrar, usando como llaves P y $h(P)$ respectivamente, los mensajes 1 y 2 de *EKE-DH* (A, $E_P(\alpha^{R_A}(mod\ \beta))$ y $E_P(\alpha^{R_B}(mod\ \beta))$, $E_k(Ch_B)$), y los mensajes 1 y 2 de *AEKE* (A, $E_{h(P)}(\alpha^{R_A}(mod\ \beta))$ y $E_{h(P)}(\alpha^{R_B}(mod\ \beta))$, $E_k(Ch_B)$), los cuales en este protocolo viajan en claro.

Las características y propiedades de *SPEKE*, que se resumen en la tabla presentada en la Figura 1 de la Subsección 5.2.7 de este Capítulo, son las siguientes:

1. **Servicios que proporciona**
   (a) *Autenticación mutua*
   (b) *Establecimiento de llave de sesión*
   (c) *Seguridad hacia adelante*
2. **Suposiciones en las que se basa**
   (a) *A y B conocen P*
3. **Ataques que Soporta**
   (a) *Diccionario*
   (b) *Texto Verificable*
   (c) *Réplica*
   (d) *Criptoanálisis sobre k*
4. **Particularidades**
   (a) *Usa intercambio Diffie-Hellman*
   (b) *No cifra mensajes 1 y 2 ($A \longrightarrow B : Q_A = F(P)^{R_A}\ (mod\ p)$ y $B \longrightarrow A : Q_B = F(P)^{R_B}\ (mod\ p)$). Usa $F(P)$ como base para exponenciación*

(c) *Usa quienvives para autenticación y saludo*

(d) *No usa mínicos*

5. **Ventajas**

(a) *No utiliza instantáneas*

(b) *No requiere intercambio previo de llave pública*

(c) *Elección bilateral de k*

(d) *La llave de sesión k no se intercambia*

(e) *A y B no acuerdan ninguna base previamente*

(f) *P es factor independiente en el proceso de autenticación*

6. **Desventajas**

(a) *Agrega un intercambio más en el saludo con respecto al protocolo EKE-DH*

De acuerdo a estas características, *SPEKE*, al igual que *EKE-DH*, cumple con todos los requerimientos básicos para este tipo de protocolos, a saber: impide ataques diccionario dentro y fuera de línea, proporciona autenticación mutua, integra el acuerdo de llave de sesión, y cumple con el requisito de que la parte A no almacene secretos persistentes. Además, cumple con los requerimientos adicionales de proporcionar seguridad perfecta hacia adelante, autenticación mutua, y no usar instantáneas. El fundamento sobre estos logros, es el mismo que para el caso de *EKE-DH*.

Comparación de SPEKE con EKE-DH

La diferencia fundamental de *SPEKE* con respecto a *EKE-DH* es que en *SPEKE* no se requiere que previamente A y B acuerden una base para Diffie-Hellman, y que mientras que *EKE-DH* y *AEKE* tienen que cifrar los mensajes 1 y 2 usando ya sea P o $h(P)$ como llave, *SPEKE* puede enviar esos mensajes en claro. En otras palabras, hace de P un factor independiente en el proceso de autenticación.

Al igual que con otros protocolos, ha habido algunos intentos de comparación entre *EKE-DH* y *SPEKE* tomando enfoques ligeramente distintos a los

mencionados en este análisis. A continuación, se describen esos intentos comparativos y sus referencias.

En [19] se examinan los protocolos *EKE-DH* y *SPEKE* a la luz de ataques ya conocidos y de nuevos ataques, junto con algunas restricciones para prevenirlos. A pesar de la similitud entre estos dos protocolos, las restricciones son distintas. También se comparan los llamados métodos fuertes con otros esquemas de autenticación, en términos de beneficios, limitaciones, y la conveniencia entre eficiencia y seguridad.

A partir del análisis realizado en [19], también se puede afirmar que los protocolos *EKE-DH* y *SPEKE* cumplen con las características fundamentales mecionadas en la subsección 5.2.1 de este capítulo. El resumen de este análisis es el siguiente.

Ambos protocolos están basados en intercambio de llave Diffie-Hellman [39] y los dos son formas de *intercambio de llave autenticada* (Sección 3.4 del Capítulo 3). Utilizan Diffie-Hellman para proteger a P de ataques diccionario y ambos protocolos usan P para evitar el ataque estándar por hombre enmedio sobre Diffie-Hellman. Los dos usan aritmética en un grupo finito muy grande [19] (pp. 4).

En *EKE-DH* y *SPEKE*, la verificación (el saludo) que A y B hacen sobre el conocimiento de la llave de sesión k es idéntico. La variación a esta etapa, mencionada en [18], consistente en usar en tres mensajes $Q_B = F(P)^{R_B}$ $(mod\ p)$ en lugar de quienvives Ch, se puede aplicar a ambos protocolos. Pero, de manera más general, puede utilizarse cualquier método clásico, ya que k es criptográficamente fuerte. En [19] (pp. 5) se propone un método de saludo que usa resultados hash de k, en lugar de núnicos aleatorios, lo cual es posible debido a que k se genera a partir de información aleatoria de ambas partes.

Al igual que *SPEKE*, *EKE-DH* puede dividirse en dos etapas. La primera etapa usa intercambio de llave Diffie-Hellman para establecer k, donde ambas partes cifran los valores exponenciales con P. Con el conocimiento de P, A y B descifran los valores exponenciales para calcular la misma k. En la segunda etapa, A y B se confirman mutuamente el conocimiento de k antes de usarla.

En [19] (pp. 6) se presenta un análisis de *EKE-DH* y *SPEKE* en términos

de restricciones necesarias a cada uno de ellos para evitar ciertos ataques por criptoanálisis como son: logaritmos discretos, cálculo de logaritmos Pohlig-Hellman, partición, confinamiento de k dentro de un subgrupo, etc. Como hemos mencionado antes, este tipo de análisis no será tratado en este trabajo debido a que las suposiciones originales sobre ataques soportados por este tipo de protocolos no considera ataques criptoanalíticos.

5.2.5 SRP-3, AEKE y B-SPEKE

El protocolo *SRP*-3 [11] no se analizó en el capítulo 4 de este trabajo debido a que se basa en técnicas de conocimiento cero cuyo conocimiento, dominio, y análisis quedan fuera de los alcances y objetivos del presente trabajo. No obstante, en esta subsección se hace un análisis muy ligero de este protocolo y, a partir de ahi, una comparación con los métodos extendidos *AEKE* y *B-SPEKE*.

SRP-3

SRP-3 combina técnicas de conocimiento cero con protocolos de intercambio de llave asimétrica y ofrece una eficiencia significativamente mejorada con respecto a los métodos extendidos, tales como *AEKE* o *B-SPEKE*, los cuales son resistentes a la eventualidad de comprometer el verificador [11] (pp. 1).

Se demuestra [11](pp. 21) que *SRP*-3 posee las siguientes propiedades:

1. Un atacante sin conocer P ni el archivo de passwords, no puede montar un ataque diccionario sobre P. En este escenario se logra autenticación mutua
2. Un atacante que obtiene el archivo de passwords no puede comprometer directamente la autenticación usuario-servidor y ganar acceso al servidor sin realizar una costosa búsqueda tipo diccionario
3. Un atacante que captura la llave de sesión no puede usarla para montar un ataque diccionario sobre P

4. Un atacante que captura P no puede usarlo para comprometer llaves de sesión de sesiones pasadas

Se cree que este conjunto de propiedades es, o está muy cerca de ser, el límite teórico que puede ofrecer un protocolo puramente basado en passwords [11] (pp. 21).

Las propiedades de *SRP*-3 se pueden resumir como sigue: resiste ataques diccionario montados por atacantes pasivos o activos, permitiendo, en principio, el uso de passwords débiles de manera segura. Ofrece seguridad perfecta hacia adelante, y los passwords se almacenan de tal modo que no contienen texto en claro equivalente al password mismo. De esta manera, un atacante que captura el archivo de passwords no puede usarlo directamente para comprometer la seguridad del protocolo y ganar acceso inmediato al servidor [11] (pp. 1).

SRP-3 basa su seguridad en la dificultad de calcular logaritmos discretos [3] módulo un número primo grande y seguro; cumple los anteriores requerimientos usando sólo un intercambio exponencial de llave, lo que lo hace útil para aplicaciones en las que la eficiencia es una necesidad. Esto resuelve algunos problemas sobresalientes en *EKE* y *SPEKE*, sin sacrificar eficiencia o seguridad. Su seguridad, simplicidad y rapidez lo hacen ideal para un amplio rango de aplicaciones del mundo real en las cuales se requiere autenticación segura basada en passwords [11] (pp. 21).

Comparación de SRP-3 con AEKE y B-SPEKE

De acuerdo a su autor, la seguridad que ofrece *SRP*-3 sólo es comparable a la de los métodos llamados extendidos (ver Figura 1), de los cuales los dos mejores ejemplos son *AEKE* y *B-SPEKE*. Estos métodos extendidos toman un protocolo simétrico basado en password (no basado en verificador) y agregan una ronda adicional al protocolo asimétrico. Por ejemplo, la forma general de *AEKE* puede describirse como sigue [11] (pp. 18):

1. A y B intercambian exponentes cifrados usando P como llave

[3] Subsección A.11.2 del Apéndice A

2. A envía a B una copia cifrada de su llave de sesión k firmada con P

3. A y B calculan k y verifican que el resultado del cálculo coincida

B-SPEKE tiene la siguiente forma general:

1. A y B intercambian exponentes usando P como una base para la exponenciación

2. B envía a A otro exponente como parte de una ronda adicional para un acuerdo de llave ElGamal

3. A y B calculan k y verifican que coincidan

En contraste, los protocolos de la familia *SRP* no son métodos extendidos debido a que P se usa directamente en el cálculo de k, eliminando de esta forma la necesidad de otra ronda para distinguir entre el conocimiento del verificador y el conocimiento de P mismo.

La forma general del protocolo *SRP*-3 es como sigue:

1. A y B intercambian residuos exponenciales aleatorios sin cifrar

2. A y B calculan k y verifican que ambos cálculos coincidan

De esta manera, eliminando los mensajes extras y la segunda ronda de exponenciación, *SRP*-3 es dos veces tan rápido como una implementación comparable de *B-SPEKE* o *AEKE* (usando el más eficiente esquema de firma digital) [11] (pp. 18).

Implementaciones prototipo de *SRP*-3 sobre una variedad de plataformas y procesadores, incluyendo *Intel x86 (i486DX2/66) y SPARC*, típicamente requieren menos de 1 seg. para autenticar usuarios, aún usando módulos tan grandes como 1024 bits, [11] (pp. 18).

5.2.6 OKE y EKE

Al igual que el protocolo *SRP*-3 [11], el protocolo *OKE* [41] no se analizó en el capítulo 4 de este trabajo debido a que, aunque utiliza esquemas de llave pública, la llave pública viaja en claro y tampoco se cifra la llave de sesión. *OKE* utiliza relaciones matemáticas para lograr sus objetivos que son basicamente intercambio de llave y autenticación. No obstante, en esta subsección se hace mención a este protocolo comparándolo con el protocolo *EKE*.

OKE

El protocolo *OKE* también utiliza esquemas de llave pública, pero las llaves se envían de forma abierta, no cifrada. En contraste con *EKE*, la misma pareja de llaves, pública y privada, puede usarse para un número arbitrario de sesiones. Una variante de *OKE* basada en *RSA* se analiza con detalle en [41] (pp. 6 - 8) y los autores la encuentran eficiente y práctica.

Comparación de OKE con EKE

Una comparación que se hace en [41] (pp. 8 - 9) entre los protocolos *OKE* y *EKE*, resume en dos las propiedades de *EKE*:

1. Cada ejecución de *EKE* requiere que A genere una nueva pareja de llaves pública y privada, (K_A^p, K_A^r)
2. *EKE* depende del cifrado de la llave pública K_A^p

El impacto de la propiedad 1 de *EKE* (requiere generar una nueva pareja de llaves en cada ejecución) es que los criptosistemas asimétricos con un proceso lento de generación de llaves tienden a hacer lenta la ejecución del protocolo. Por ejemplo, el cifrado asimétrico con ElGamal y el intercambio de llaves Diffie-Hellman no requieren un proceso largo de generación de llaves, pero *RSA* sí lo requiere. *RSA* requiere seleccionar dos números primos muy

grandes. Esta es una desventaja de EKE con respecto a OKE, ya que en este último una llave generada una vez puede ser usada siempre [41] (pp. 9).

Si se considera una implementación de EKE basada en RSA, la situación se vuelve difícil debido a la propiedad 2 de EKE (depende del cifrado de la llave pública de A). Resulta difícil hallar la manera de cifrar el módulo N para RSA sin que los intentos de descifrarlo usando una llave incorrecta no revelen el hecho de que el resultado de tales intentos sean un módulo inválido para RSA. Los autores de EKE sugieren enviar el módulo N en claro y sólo cifrar el exponente público e de la pareja (e, N) que constituye la llave pública para RSA [17]. Ese exponente debe ser un número entero impar aleatoriamente seleccionado.

El punto es cómo escoger dos primos muy grandes p y q, tales que sean primos relativos a $\varphi(pq) = (p-1)(q-1)$, condición que debe cumplirse para RSA [41] (pp. 9).

El protocolo OKE con RSA permite escoger el exponente e como se desee. Ejemplo, $e = 3$ para cifrado rápido RSA. EKE con RSA requiere que e sea un número impar aleatorio, lo cual hace lento el cifrado. Por otro lado, OKE requiere que B cifre $(K + 1)$ textos en claro con RSA y que A los descifre. K es un número natural que funciona en OKE como parámetro de seguridad.

La principal ventaja de OKE con RSA en comparación a EKE con RSA, es que las llaves públicas se generan una sóla vez. Esto supera su principal desventaja que consiste en que B tiene que cifrar los $(K + 1)$ textos en claro y A tiene que descifrarlos.

La propiedad 1 de EKE (requiere generar una nueva pareja de llaves en cada ejecución) es necesaria ya que, como se ilustra en [41] (pp. 9), de no ser así, EKE sería vulnerable a ataques por texto verificable de la siguiente manera:

1. El atacante I observa la ejecución del protocolo entre A y B. Observa que A envía $x = E_P(K_A^p)$ a B en la primera ronda y que B replica con $y = E_P(E_{K_A}^p(k))$ en la segunda ronda
2. I necesita la llave de sesión k. Si A utiliza $x = E_P(K_A^p)$ otra vez, I ataca a P por diccionario en x y compara con la registrada en la

ejecución anterior.

También, si A reutiliza $K_A{}^P$ para comunicarse con otra parte C, usando un password diferente ρ, entonces I puede hallar ambos passwords, P y ρ, por fuerza bruta observando dos cantidades P_i y ρ_j, con iguales descifrados $D_{P_i}(x) = D_{\rho_j}(z)$, donde $x = E_P(K_A^p)$ y $z = E_\rho(K_A^p)$.

5.2.7 Tabla Comparativa

La Tabla presentada en la Figura 1 de esta subsección resume las características y propiedades de los protocolos fuertes analizados en el capítulo 4 de este trabajo y del análisis comparativo hecho de esos protocolos en el presente capítulo. La tabla consta de ocho columnas conteniendo la siguiente información: la columna 1 corresponde al nombre con el cual se conoce al protocolo en la literatura sobre el tema, la columna 2 contiene las referencias principales sobre el protocolo, la columna 3 enumera los servicios de seguridad que proporciona, la columna 4 menciona las suposiciones principales en las que se basa, la columna 5 menciona el tipo de ataques que soporta, la columna 6 resume las características principales que lo distinguen, la columna 7 enumera las ventajas del protocolo, y la columna 8 hace lo propio con las desventajas.

PROTOCOLOS FUERTES PARA AUTENTICACIÓN DIRECTA							
Protocolo	**Ref.**	**Proporciona**	**Suposiciones**	**Ataques**	**Características**	**Ventajas**	**Desventajas**
LGSN Original	[1] [6]	Autenticación mutua Intercambio de K_A^o y k	A y B conocen P	Diccionario Texto Verificable Réplica	P cifra K_A^o Doble cifrado mensaje 2 Usa f(Ch) para saludo y autenticación Núnicos	No instantáneas No intercambio previo de K_A^o	K_A^o secreta k se intercambia No revelar K_A^o A genera K'_A, K_A^o y k
LGSN Reforzado	[1]	Autenticación mutua Intercambio de K_A^o Establecimiento de k	A conoce P B conoce h(P)	LGSN Original Criptoanálisis sobre k	LGSN Original h(P) cifra K_A^o	LGSN Original Elección bilateral y no intercambio de k	LGSN Original
EKE Original	[7] [17]	Autenticación mutua Intercambio de K_A^o y k	A y B conocen P	Diccionario Texto Verificable Réplica	P cifra K_A^o Ch para saludo y autenticación Doble cifrado mensaje 2	B genera k No intercambio previo de K_A^o No instantáneas	k se intercambia A genera K_A^o y K'_A en cada ejecución K_A^o y K'_A secretas
EKE Reforzado	[17]	Autenticación mutua Intercambio de K_A^o Establecimiento de k	A y B conocen P	EKE original Criptoanálisis sobre k	EKE original Usa f(Ch) para Saludo y autenticación No usa núnicos	No intercambio previo de K_A^o Elección bilateral k No intercambio de k	EKE original Complejidad
EKE - DH	[17] [18] [19]	EKE Reforzado Seguridad hacia adelante	A y B conocen P A y B acuerdan α y β	EKE Original Criptoanálisis sobre k	Usa Diffie-Hellman P cifra mensajes 1, 2 Ch para saludo y autenticación	Elección bilateral k No intercambio previo de K_A^o No intercambio k	Dificultad en acuerdo previo de α y β
AEKE	[7] [17]	EKE Reforzado Seguridad hacia adelante	A conoce P B conoce h(P) A y B acuerdan α y β	EKE Original Compromiso de h(P) Criptoanálisis sobre k	Usa Diffie-Hellman Ch para saludo y autenticación f(P, k) protege compromisos de h(P)	Resiste compromiso de h(P). Elección bilateral y no intercambio k No suplantación de A ante B	Agrega un intercambio extra Suplantación de B ante A Dificultad en acuerdo previo de α y β
SPEKE	[19]	Autenticación mutua Establecimiento de k Seg. hacia adelante	A y B conocen P	EKE Original Criptoanálisis sobre k	Usa Diffie-Hellman. Usa f(P) como base Ch para saludo y autenticación	Elección bilateral y no intercambio k No intercambio K^o P es factor independiente	Agrega un intercambio más
B-SPEKE	[20]	Autenticación mutua Establecimiento de k Seg. hacia adelante	A conoce P B conoce $V_1 = g^P$	Diccionario Texto Verificable Réplica	Dos intercambios Diffie-Hellman Basado en verificador	Elección bilateral de k P es factor independiente	Agrega un intercambio más

Figura 1: Tabla Comparativa de Protocolos para Autenticación Fuerte

5.3 Evolución, Influencias y Derivaciones

Con la aparición de la primera propuesta de solución hecha por Gong, Lomas, Saltzer y Needham en 1989 [6], se inició una larga lista de trabajos relacionados influenciados, de una u otra manera, por esta propuesta y por las versiones hechas por los mismos autores en [1], aparecidas en 1993. Estas últimas, mantienen la estructura básica del *LGSN* Original, y se pueden resumir como sigue:

1. *LGSN Reducido*

   Esta versión se presenta en la subsección 4.1.2 del capítulo 4 de este trabajo y consiste básicamente en reducir el número de mensajes que se intercambian en el protocolo. Esta reducción se hace debido que el *LGSN* Original es un protocolo sólo para demostración del método de solución, sin considerar ningún tipo de optimización.

2. *LGSN con Núnicos*

   Esta versión se presenta en la subsección 4.1.3 del capítulo 4 de este trabajo y consiste en eliminar las instantáneas en el *LGSN* Original, partiendo de la consideración de que no siempre se puede asegurar su utilización en todos los sistemas, y debido a los problemas de sincronización inherentes a su uso.

3. *LGSN con Distribución de Llaves Públicas*

   Esta versión de *LGSN* se presenta en la subsección 4.1.4 del capítulo 4 de este trabajo y consiste en integrar al protocolo la distribución de las llaves públicas de A y de B, basándose para ello en la versión Reducida de *LGSN*.

4. *LGSN para Autenticación Directa*

   Esta versión de *LGSN* se presenta en la subsecćon 4.1.5 del capítulo 4 de este trabajo y consiste en eliminar la suposición de que existe un servidor de llaves S, de tal modo que en esta versión solamente participan A y B en el proceso de autenticación. Esta versión asume que A y B conocen P.

5. *LGSN Reforzado para Autenticación Directa*

Esta versión, presentada en la Secc. 4.1.6 del capítulo 4 de este trabajo, refuerza la versión anterior al suponer que sólo A conoce P y B conoce $h(P)$. En esta versión, tanto A como B participan en la elección de k, lo cual evita que k se intercambie por el canal, evitando ataques por criptoanálisis sobre k.

6. *LGSN Óptimo para Autenticación Directa*

   Esta versión, presentada en la subsección 4.1.7 del capítulo 4 de este trabajo, optimiza las dos versiones anteriores en términos de número de mensajes y rondas. En este protocolo el número de mensajes y rondas es de tres, siendo este el límite inferior probado para protocolos basados en núnicos [10].

Sobre esta misma línea de influencia, también en 1993, aparece el trabajo [36] de Tsudik y Van Herreweghen donde se exponen algunos problemas de los protocolos $LGSN$, tales como la utilización de instantáneas y la necesidad del servidor de almacenar información de estado. En ese trabajo se presentan versiones más simples y eficientes, respetando la estructura básica de los originales.

En 1995, L. Gong presenta un nuevo trabajo [55] sobre los protocolos $LGSN$, pero ahora enfocando su análisis a lograr la optimización de ellos en términos de número de mensajes y rondas. En ese trabajo se presentan versiones optimizadas de los protocolos $LGSN$.

Se puede decir que el desarrollo de los protocolos $LGSN$ termina, o se detiene, en 1995 con el trabajo [55] de L. Gong, reuniendo hasta esa fecha un conjunto de protocolos fuertes viables basados en passwords. Actualmente estos protocolos se encuentran en una etapa de evaluación y estudio de aplicaciones para eventualemnte implementarlos en tecnologías emergentes como Java [63]. $NetWare$ versión 4 [67] tiene implementado un protocolo similar al $LGSN$ Original para que el usuario de la red pueda bajar su llave privada cifrada de un directorio y poder abrir una sesión segura en esa red [68]. A través de estas implementaciones, estos protocolos pueden lograr su consolidación como soluciones fuertes, al poder ser comparadas con otras soluciones igualmente viables, como son los protocolos de la gran familia EKE, y otras soluciones basadas en conocimiento cero.

Aunque no han aparecido recientemente nuevas versiones o derivaciones directas de los protocolos *LGSN*, su influencia en protocolos de reciente aparición es evidente, como es el caso de los protocolos Halevi-Crawczyk [61] que toman como influencia de *LGSN* el uso de técnicas de llave pública.

También los protocolos conocidos como *S3P* [58], los cuales basan sus propiedades completamente en sistemas de llave pública, se ven influenciados por los protocolos LGSN al usar el concepto de *confundidores*.

En la otra gran familia de protocolos basados en passwords conocida como *EKE* [17, 7], propuestos por Bellovin y Merrit en 1992, también aparece la influencia de los protocolos *LGSN* a través del uso combinado de técnicas criptográficas de llave secreta y de llave pública.

A partir de 1992, la familia *EKE* forma su propia rama de derivaciones e influencias, como se muestra en la Figura 2, siendo la versión de *EKE* que utiliza intercambio Diffie-Hellman, conocida como *EKE-DH* [17], la base principal de esa evolución. Esta versión conserva la estructura básica del protocolo *EKE* original.

En 1994 aparece una derivación de *EKE-DH* conocida como *AEKE* [7] que se distingue por usar un verificador, puede ser $h(P)$, en B que es utilizado como prueba de que A conoce P. La idea de esta versión es proteger los passwords contra ataques diccionario y contra compromiso del archivo donde se guardan los verificadores. Esta versión hereda del protocolo *EKE-DH* el intercambio Diffie-Hellman.

También, tomando como base el protocolo *EKE-DH*, en 1996 aparece el protocolo conocido como *SPEKE* [19] que toma como influencia de *EKE-DH* el intercambio Diffie-Hellman, pero usa P para construir una base adecuada para la exponenciación en dicho intercambio.

A partir de *SPEKE* surgen algunos de los métodos conocidos como extendidos. Se conocen como extendidos porque extienden los protocolos *EKE-DH*, *SPEKE* y *AEKE*. Estos métodos son los siguientes. El protocolo *A-SPEKE*, el cual también toma influencia de *AEKE* ya que consiste básicamente en una aplicación de este a *SPEKE*. El protocolo *B-SPEKE*, el cual asume que ambas partes tienen acceso a un verificador que usa P como exponente, y que solamente A conoce P; utiliza dos intercambios Diffie-Hellman: el primero usa *SPEKE* para derivar una llave

temporal basada en $h(P)$, el segundo genera otra llave temporal usando P como exponente. El protocolo *B-EKE*, el cual resulta de reemplazar, en *B-SPEKE*, a *SPEKE* con *EKE-DH*.

Los métodos extendidos buscan mejorar el rendimiento de *EKE* y *SPEKE* al hacer de P un factor independiente en el proceso de autenticación. Estos métodos tienen vigencia actual y constituyen parte del estado del arte en soluciones de este tipo a la fecha, como se muestra en la Figura 2. La influencia de *EKE* y *SPEKE* también se ha hecho presente en recientes trabajos sobre protocolos para bajar una llave privada de un directorio con el fin de que un usuario A pueda darse *"login"* en una red [68].

Por otro lado, el protocolo *EKE-DH* también influye con su intercambio asimétrico en el protocolo no criptográfico *SRP* [11], del cual deriva como versión el protocolo *SRP*-3 [11], ambos en el estado del arte actual en cuanto a protocolos basados en passwords, aunque no se basan en criptografía, sino en relaciones matemáticas y técnicas de conocimiento cero.

Igualmente, la influencia del protocolo *EKE* [17] está también presente con sus esquemas de llave pública en el protocolo conocido como *OKE* [41], aunque en este último protocolo las llaves públicas no se cifran. *OKE* a su vez influye con esos mismos esquemas en los protocolos genéricamente conocidos como *S3P* [58].

Todo lo que se ha descrito en esta sección en cuanto a evolución, derivaciones e influencias de protocolos basados en passwords, se resume en la estructura arborescente de la Figura 2, donde se muestra el estado del arte, a la fecha, de este tipo de protocolos. Esta evolución muestra la relativa juventud de esta área, de 1989 a la fecha, área que sin duda seguirá desarrollándose y buscando soluciones cada vez más seguras y eficientes al problema de autenticación basada en passwords.

5.4 Implementaciones Existentes

En esta sección se describe el estado actual respecto a las implementaciones existentes de protocolos fuertes tanto a nivel de prototipo como comerciales, y de propósito especial, entre otras; también se mencionan las espectativas

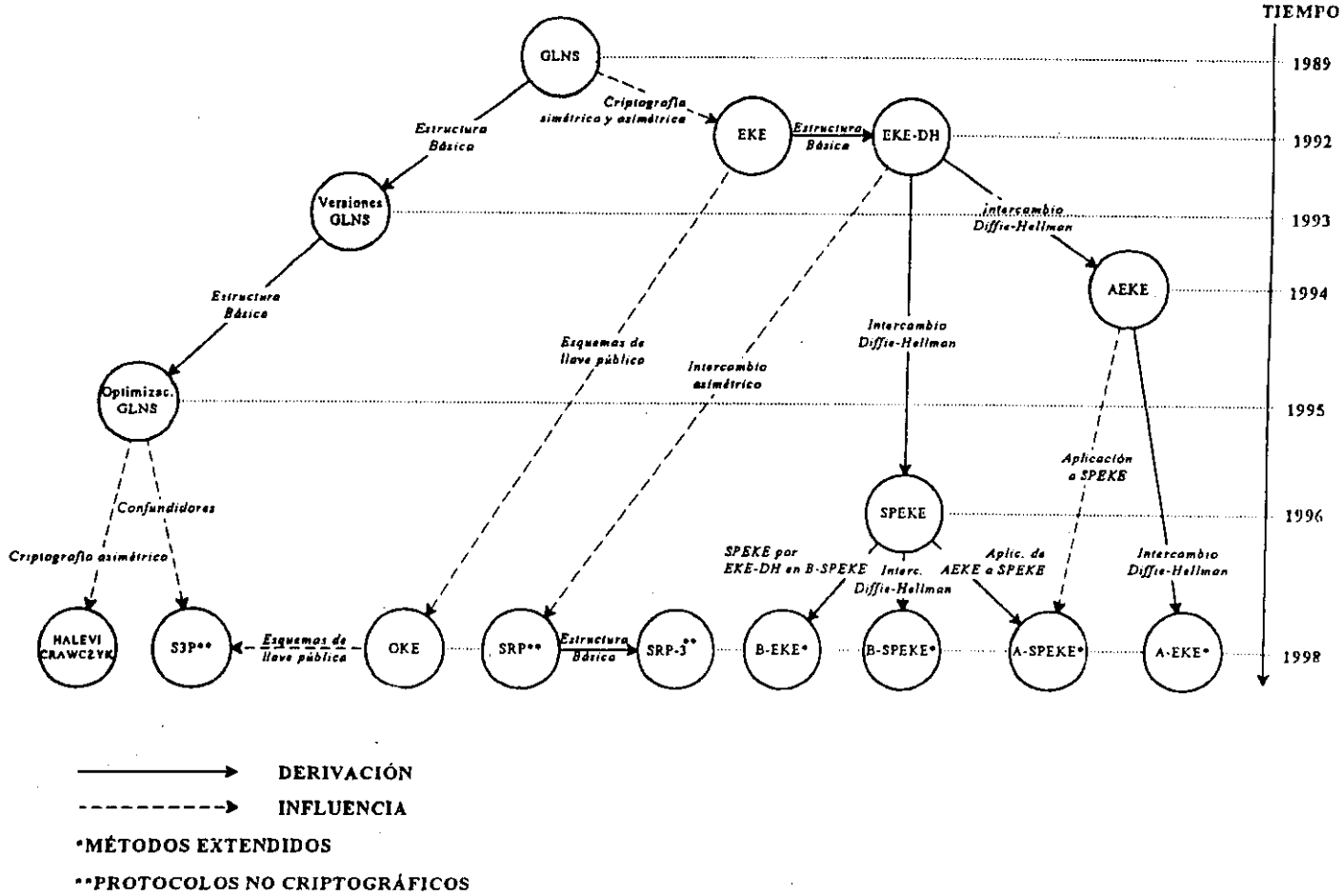

FIGURA 2.- EVOLUCIÓN DE PROTOCOLOS DE AUTENTICACIÓN FUERTE BASADOS EN PASSWORDS

acerca de implementaciones futuras de este tipo de protocolos.

La compañía *Integrity Sciences, Inc.* [52] dedicada al diseño y desarrollo de sistemas seguros para computadoras personales, localizada en Westboro, Massachusetts, USA. y fundada en 1992 por David Jablon, ha desarrollado implementaciones de *SPEKE* y *B-SPEKE* [19, 20]. Es posible adquirir una licencia para fines particulares y comerciales o disponer de ellas sin costo para fines no lucrativos en la dirección electrónica que aparece en [52] con sólo solicitarla.

Se sabe [11] (pp. 18) de implementaciones prototipo en la Universidad de Stanford del protocolo *SRP*-3 sobre varias plataformas y procesadores, incluyendo *Intel x86 (i486DX2/66) y SPARC* para fines de comparación de rendimientos con respecto a *AEKE* y *B-SPEKE*, principalmente.

En comunicación personal con Steve Bellovin [62], coautor de *EKE* y *AEKE* [7, 17], él ha dicho: *"las únicas implementaciones de EKE y AEKE de que tengo conocimiento fueron las realizadas para un proyecto interno de la compañía ATT"*.

De igual forma, en correspondencia personal con Li Gong [63], coautor de la familia de protocolos *LGSN* [1, 6], él ha mencionado no tener información sobre implementaciones prácticas. También, Gong cree que algunos productos Java, en el futuro, pueden contener tales implementaciones.

En comunicación personal con Sarvar Patel [64], autor de un trabajo [60] donde presenta ataques por diccionario, usando resultados de la teoría de números, sobre todas las versiones de *EKE* discutidas en [17], él menciona que es sorprendente que no se tenga conocimiento de implementaciones de estos protocolos, como se tienen de otros métodos de passwords tales como *S-Key* y SecureID. Cree, no obstante, que se ha mencionado la idea de proponer estos protocolos como *RFC's*(*"Request For Comments"*) para *Internet*.

En publicaciones recientes [69], que analizan debilidades en los actuales esquemas de seguridad en red basados en passwords, tales como Kerberos, se propone y recomienda fortalecer tales esquemas usando protocolos fuertes como *EKE*, *SPEKE*, y *SRP*.

5.5 Eficiencia de Métodos Fuertes

En esta sección se describe lo que se conoce actualmente respecto a eficiencia de este tipo de protocolos, los estudios que se han hecho, y su comparación respecto a la eficiencia de protocolos clásicos basados en passwords.

Debido a que no existen aún implementaciones ampliamente difundidas de estos protocolos, y de las que se sabe que se han hecho [11] (pp. 18) no se tienen estadísticas de eficiencia, es difícil conocer con certeza su rendimiento.

De los pocos autores que tocan el tema de la eficiencia de estos protocolos es Thomas Wu [11] de la Universidad de Stanford donde se han realizado implementaciones prototipo de protocolos fuertes y se han realizado algunas pruebas de eficiencia.

En [11] (pp. 3) se afirma que el pobre rendimiento que logran los protocolos llamados fuertes, ha sido con frecuencia un obstáculo para adoptarlos como base en implementaciones concretas. Los protocolos descritos en [7] y [20] son demasiado lentos para necesidades de autenticación frecuentes y rápidas. Una mejora de 3 a 1.5 segundos en el rendimiento puede hacer la diferencia entre una solución intolerable y una aceptable.

En [11] (pp. 17), T. Wu afirma: *"cualquier mejora del tiempo de ejecución (de este tipo de protocolos) necesitaría conmutar a otra construcción distinta de la basada en exponenciación discreta. Una opción prometedora son los criptosistemas basados en curvas elípticas, los cuales pueden ofrecer potencialmente el mismo nivel de seguridad que los basados en la dificultad de logaritmos discretos o factorización, pero con llaves mucho más cortas. Una implementación de AKE ("*Asymetric Key Exchange*") [11] (pp. 3 - 4), el cual es una forma generalizada para una clase de protocolos basados en verificador, con curvas elípticas podría esperarse que se ejecutara varias veces más rápido que SRP-3, proporcionando el mismo nivel de seguridad. Es aún demasiado pronto para hacer cualquier pronunciamiento firme sobre la seguridad de las curvas elípticas, debido a que no han sido analizadas tan extensamente como los exponentes discretos. Además, las curvas elípticas están limitadas por derechos de autor y restricciones de patente, el cual no es el caso para exponenciación discreta simple. Opino que SRP-3 es bastante eficiente, aún sobre el hardware actual, de tal manera que el rendimiento no es un problema significativo. En la mayoría de casos, el tiempo requerido*

para negociar el protocolo de autenticación no es aún notable para el usuario, y esto sólo mejorará conforme el hardware se hace más rápido. La seguridad proporcionada por el problema de logaritmos discretos y el problema de factorización, satisface aún los más conservadores requerimientos de seguridad".

5.6 Aplicaciones Actuales y Futuras

Esta sección describe la amplia gama de ámbitos y aplicaciones en las que este tipo de protocolos pueden ser de gran utilidad. Estas aplicaciones van desde sustitución de alternativas actualmente inseguras y en uso, hasta telefonía celular y sistemas incrustados.

En [17] (pp. 10 - 11) se menciona la autenticación de un usuario ante un host como motivación para el diseño de EKE; pero también se mencionan otros usos posibles, de los cuales los más interesantes parecen ser la aplicación de EKE en seguridad de teléfonos públicos, y de teléfonos celulares para evitar fraudes y asegurar la confidencialidad de la llamada.

Igualmente en [17] (pp. 10 - 11) se menciona la utilización de EKE para reemplazar el protocolo de Rivest y Shamir llamado de Interbloqueo (*"Interlock Protocol"*) [38] (pp. 54 - 55) originalmente diseñado para detectar intrusos activos y posteriormente utilizado para autenticación, pero susceptible a ataques por hombre enmedio mostrado en [56]. La razón de esta propuesta de reemplazo es que el ataque ahí descrito no funciona contra EKE. También se mencionan la posible utilización de EKE como un amplificador de privacía, ya que puede usarse para fortalecer sistemas simétricos y asimétricos que son comparativamente débiles cuando se usan juntos.

Los métodos fuertes son ideales en algunas aplicaciones por varias razones: el almacenamiento seguro de llaves puede ser problemático, tener llaves grandes de entrada puede se molesto, los passwords pueden requerir mejor protección, entre otras.

En [19] se discute el uso de $EKE\text{-}DH$ y $SPEKE$ para corregir deficiencias en Kerberos. Las mismas ideas pueden usarse para mejorar muchos otros sistemas de *"login"* en red, los cuales pueden ser esenciales para nuevos

usos, tales como servicios bancarios remotos a través de una computadora personal, y acceso general a computadoras sobre Internet.

En sistemas donde solamente se dispone de una clave numérica, tal como autenticación en telefonía celular, es especialmente conveniente un password numérico corto y seguro. Estos ambientes pueden evitar el almacenamiento a largo plazo de llaves persistentes.

Las estaciones de trabajo sin disco son otra clase de dispositivos donde es inconveniente tener llaves almacenadas localmente. Los métodos basados en passwords son ideales para establecer una conexión inicial a un servidor seguro, y obtener seguramente las credenciales almacenadas de los usuarios. Este concepto de almacenamiento remoto de credenciales de largo plazo con una bajada segura ha sido usado en el sistema de autenticación *SPX* [53].

Cualquier dispositivo o sistema que use estos métodos puede ser genérico, en el sentido de no estar programado para comunicarse solamente con un usuario o host en particular. Los enfoques alternos que usan una llave pública fija o persistente necesitan que esa llave fija sea asignada por una autoridad confiable específica. Esto puede ser un inconveniente. Los métodos basados en passwords pueden hacer más fácil desarrollar un sistema que no tenga tales restricciones.

A pesar que estos métodos se enfocan actualmente a la autenticación de un usuario ante un host, pueden ser importantes para autenticación usuario-usuario. En [57] se describe el uso de una serie interactiva de preguntas y respuestas para autenticar la identidad de alguien conocido a través de una red. La idea es que cada uno pruebe al otro que conocen hechos comunes sin revelar esos hechos. Tales situaciones son actualmente muy comunes. Los métodos basados en passwords resuelven esta paradoja general de autenticación, la cual se presenta aún en algunas situaciones frente a frente. Por ejemplo, se desea probar que un banco conoce un número de cuenta secreto, y el banco desea que se pruebe conocer el número secreto de seguro social, pero nadie quiere revelar directamente la información al otro.

Capítulo 6

Resultados, Conclusiones y Trabajo Futuro

En este capítulo se hace un resumen de los logros alcanzados en este trabajo con respecto a resultados obtenidos, y las conclusiones que se desprenden de esos resultados. También, se mencionan algunas de las principales tareas que quedan pendientes de realizar en un futuro próximo, tomando como punto de partida este trabajo.

6.1 Resultados

En este trabajo se ha presentado un estudio sobre los protocolos criptográficos que se basan en passwords para lograr autenticación e interambio de llaves de manera segura y confidencial, sobre una red de comunicaciones insegura. Este estudio muestra los principales ataques a este tipo de protocolos, y las fortalezas y debilidades que cada uno de ellos presenta a esos ataques. Se han presentado igualmente, las características básicas que este tipo de protocolos debe cumplir y algunas otras características que idealmente deberían cumplir.

El estudio de referencia se ha hecho sobre la base de identificar y definir con presición el problema, el modelo, los tipos de ataques, las suposiciones y los

objetivos concretos en cada protocolo analizado. Aunque el análisis se hizo en general para este tipo de protocolos, se han enfatizado aquellos protocolos de autenticación mutua y directa de dos partes que utilizan criptografía para lograr sus objetivos de seguridad. Este énfasis se debe a que en este esquema no se requiere una tercera parte confiable tal como un servidor de llaves, y por lo mismo la implementación puede ser más sencilla y económica.

Se han analizado las principales corrientes de investigación que se siguen para buscar mejores soluciones actualmente; se han dado ejemplos concretos de protocolos de cada corriente, y se ha mostrado dónde se ubica el estado del arte en estas direcciones.

Para hacer posible lo anterior, se han estudiado cuarenta y nueve artículos originales sobre este tipo de protocolos, seis libros especializados [38, 45, 47, 49, 50, 51], nueve artículos sobre estándares y criptografía [5, 8, 14, 21, 28, 29, 31, 39, 66]. Se han consultado dos direcciones electrónicas sobre protocolos fuertes y seguridad en el Web respectivamente [52, 54], y se han mantenido comunicaciones directas con algunos de los autores [62, 63, 64] de las principales familias de protocolos, como son LGSN [1, 6] y EKE [7, 17].

Como contribución fundamental de este trabajo en el área de protocolos fuertes de autenticación e intercambio de llaves basados en passwords, se hizo un análisis comparativo de los protocolos presentados en el capítulo 4. Este análisis se presenta en la Sección 5.2 del Capítulo 5 y se resume en la tabla de la Figura 1 de este trabajo; fue hecho en función de sus propiedades, suposiciones, características, ataques que soportan, ventajas, y desventajas. Hasta el momento, no se tiene conocimiento de la existencia de un análisis comparativo similar que muestre las características de cada protocolo comparándolas con las de otros protocolos del mismo tipo para poder apreciar sus diferencias, ventajas, y desventajas de cada uno con respecto a los demás.

El análisis mencionado puede resultar de vital importancia si se considera que la información sobre estos protocolos está muy dispersa y no se tiene una idea clara y concreta de las soluciones fuertes que existen para esquemas basados en passwords. Esta dispersión en la información también se debe al poco tiempo que estas soluciones tienen de haberse conocido y empezado a estudiar en serio. Antes de 1990 se creía que el problema de autenticación fuerte usando passwords débiles podía no tener solución. Otro factor es que casi no existen implementaciones de estos protocolos que no sean prototipos de estudio en universidades y para proyectos especiales.

Otra contribución de este trabajo es un análisis del desarrollo y evolución en el tiempo de estos protocolos, sus influencias y derivaciones hasta el momento. Este análisis se presenta en la Sección 5.3 del Capítulo 5 y se resume en una estructura arborescente que se presenta en la Figura 2 de este trabajo. Un trabajo similar a esta clasificación se presenta en [52], pero no incluye todos los protocolos presentados en este trabajo y no menciona influencias específicas.

Otro aspecto importante de este trabajo lo constituye la búsqueda de información y referencias acerca de implementaciones concretas, ya sean comerciales, académicas, prototipos para estudio, o proyectos especiales que se hubieran realizado de protocolos fuertes. El resultado de esta investigación se presenta en la Sección 5.4 del Capítulo 5 de este trabajo. De la misma manera, se trató de recabar la mayor información posible sobre pruebas y estudios de eficiencia de estos protocolos, que se hubieran realizado tanto a nivel teórico como en implementaciones prototipo para analizar su rendimiento. Los resultados de esta búsqueda se presentan en la Sección 5.5 del Capítulo 5 de este trabajo.

Finalmente, se mencionan los distintos ámbitos y aplicaciones en los que este tipo de protocolos pueden ser implementados, mencionando sus ventajas con respecto a lo que como alternativa de solución se tiene en la actualidad, y las ventajas que significarían cambiar esas implementaciones actuales por soluciones fuertes. Estos ambientes y aplicaciones se presentan en la Sección 5.6 del Capítulo 5 de este trabajo.

6.2 Conclusiones

Una conclusión sobre este trabajo es que se ha mostrado con amplitud y detalle la posibilidad de lograr autenticación e intercambio de llaves seguros basándose solamente en passwords débiles, algo que antes de 1990 se creía no solamente muy difícil, sino imposible de lograr. Esta certeza de poder contar con soluciones viables a este problema es de suma importancia debido a la gran proliferación de esquemas de autenticación que se basan en passwords, en distintos ámbitos y aplicaciones. Lo anterior también es importante por la sencillez que representan estos métodos que sólo requieren de la parte que se autentica un pequeño password memorizado, algo que parece muy

natural y fácil de lograr. Estos métodos tienen la ventaja adicional de que son relativamente fáciles de implementar y poco costosos en relación a otros métodos que no se basan en passwords.

Una conclusión más de este trabajo es la evidente juventud del área de protocolos fuertes basados en passwords dentro de los protocolos de autenticación en general. Esta evidencia es notoria cuando se observa que la información sobre ellos es muy escasa y dispersa, que no existen estudios comparativos de características, diferencias, ventajas, desventajas, eficiencia, etc. En este sentido, este trabajo es pionero.

Otra conclusión importante de este trabajo es que no todo está dicho y resuelto en esta área. Existen aún direcciones abiertas de investigación que buscan nuevas formas de resolver el mismo problema con mayor sencillez, seguridad y eficiencia. Constantemente están apareciendo trabajos con contribuciones importantes en el área, y modificaciones a los resultados que ya existen. No obstante, a la mayoría de estos métodos les falta también pasar la prueba de fuego que significan las implementaciones a gran escala en ambientes abiertos tales como sistemas operativos clásicos, distribuidos, y de red; sistemas para transacciones electrónicas bancarias y comerciales; telefonía celular; etc. Lo que resulta muy motivante, es el creciente interés en este tipo de protocolos y su posible implementación en un futuro cercano en tecnologías de código distribuible como Java.

Aunque estos protocolos se han venido estudiando y analizando con cuidado a últimas fechas, la historia ha probado que soluciones que se creían seguras fueron halladas con serias debilidades cuando se implementaron en sistemas para utilización masiva. Esto se debe, en mucho, a que no existe aún manera de probar a priori, con absoluta certeza, si un protocolo es seguro o no. Este es un problema no trivial de resolver y constituye una área abierta de investigación en nuestros días.

Finalmente se puede afirmar, a partir de este trabajo, que este tipo de protocolos ha tenido, tiene, y con seguridad seguirá teniendo, una amplia gama de aplicaciones, y que resolverá, en su momento, deficiencias de seguridad que ahora constituyen un gran problema en gran cantidad de ámbitos. Las aplicacion es de estos métodos van desde acceso a sistemas de cómputo, hasta cajeros automáticos, teléfonos celulares, tarjetas inteligentes y aplicaciones sobre *Internet*, entre las más conocidas. Todas estas aplicaciones, contextualizadas desde una perspectiva de servicios seguros que requiere la

sociedad actual, permiten pensar en la necesidad de conocer a fondo estos métodos, su estudio y análisis. Esto permitirá, entre otras cosas, estar en condiciones de elegir la mejor solución de acuerdo a las necesidades de cada caso en particular.

6.3 Trabajo Futuro

A partir de este trabajo, se pueden vislumbrar tareas importantes a realizar para seguir la investigación en esta dirección. Algunas de estas posibles tareas son las siguientes:

1. Como consecuencia y resultado adicional de este trabajo se espera poder escribir al menos dos artículos arbitrados: uno que muestre el análisis comparativo aquí presentado entre los protocolos fuertes de autenticación, y otro que constituya una propuesta propia y específica de solución fuerte.

2. Un trabajo pendiente de hacer es analizar más a fondo protocolos no criptográficos tales como *OKE* [41] y de la familia *SRP* del cual *SRP*-3 [11] es un representante. Estos protocolos requieren un estudio detallado sobre técnicas de conocimiento cero y matemáticas en general, ya que ambos las utilizan para lograr que los intercambios de mensajes entre las partes viajen en claro.

3. Un trabajo interesante por hacer es la implementación local de alguna o algunas de las soluciones estudiadas, analizando su seguridad y eficiencia, entre otras cosas. Esta implementación puede enmarcarse dentro de un proyecto interdisciplinario que involucre algún ámbito local específico que requiera autenticación fuerte, así como también la colaboración de programadores e investigadores interesados en un proyecto de este tipo.

4. A partir de lo estudiado, se pueden proponer soluciones que busquen integrar las mejores características de los protocolos analizados, y que de esta idea puedan surgir publicaciones de alto nivel en esta área.

5. Una tarea más ambiciosa sería trabajar en la búsqueda de una herramienta formal para poder probar con certeza si un protocolo de

autenticación dado es seguro o no, antes de intentar su posible implementación. Hasta el momento, en esta dirección existen intentos interesantes, pero que aún logran resultados limitados y parciales.

Referencias

[1] L. Gong, T. M. A. Lomas, R. M. Needham, y J.H. Saltzer, "Protecting Poorly Chosen Secrets from Guessing Attacks," *IEEE Journal on Selected Areas in Communications*, Vol.11, No.5, pp.648 - 656, Junio, 1993.

[2] R. Morris y K. Thompson, "Password Security: A Case History," *Communications of the ACM*, Vol.22, No.11, pp. 594 - 597, Noviembre, 1979.

[3] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva y M. Yung, "Systematic design of two-party authentication protocols," *Advances in Cryptology*, Proceedings of CRYPTO 91, Springer-Verlag, 1991.

[4] J.G. Steiner, C. Neuman, y J.I. Schiller, "Kerberos: An Authentication Service for Open Networks Systems," *Proceedings of the USENIX Winter Conference*, pp. 191 - 202, Febrero, 1988.

[5] U.S. National Bureau of Standards, "Passwords Usage," *U.S. Federal Information Precessing Standards Publications*, FIPS PUB 112, Mayo, 1985.

[6] L. Gong, T.M.A. Lomas, R.M. Needham, y J.H. Saltzer, "Reducing Risks from Poorly Chosen Keys," *Proceedings of the 12th ACM Symposium on Operating Systems Principles*, Litchfield Park, Arizona, Diciembre, 1989. Publicado como *ACM Operating Systems Review*, Vol.23, No.5, pp. 14 - 18.

[7] S. M. Bellovin y M. Merritt, "Augmented Encrypted Key Exchange: A Password-Based Protocols Secure Against Dictionary Attacks and

Password File Compromise," *First ACM Conference on Computer and Communications Security*, Noviembre, 1993.

[8] "PIN Manual: A Guide to the Use of Personal Identification Numbers in Interchange," *MasterCard International*, Inc., Septiembre, 1980. Reimpreso in *Cryptography: A New Dimension in Computer Data Security*, por Meyer, C.H. and Matyas, S.M., John Wiley and Sons., pp. 429 - 444, 1982.

[9] L. Gong y P. Syverson, "Fail-Stop Protocols: An Approach to Designing Secure Protocols," *Pre-proceedings of DCCA-5: Fifth International Working Conference on Dependable Computing for Critical Applications*, pp. 44 - 55, Urban-Champaign, Illinois, Septiembre, 1995.

[10] L. Gong, "Efficient Network Authentication Protocols: Lower Bounds and Optimal Implementations," *Distributed Computing*, Vol.9, No.3, pp. 131 - 145, 1995.

[11] T. Wu, "The Secure Remote Password Protocol," *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, San Diego, Marzo 1998, pp. 97 - 111.

[12] N. Haller y R. Atkinson, "On Internet Authentication," RFC 1704, *Bell Communications Research and Naval Research Laboratory*, Octubre 1994.

[13] D.V. Klein, "Foiling the Cracker: A Survey of, and Improvements to, Password Security," *Proceedings of the USENIX Security Workshop*, Invierno, 1990.

[14] National Bureau of Standards, "DES Modes of Operation," *Federal Information Processing Standard*, U.S. Department of Commerce, FIPS PUB 81, Washington, D.C., 1980.

[15] W. Belgers, "UNIX Password Security," Report of Technical University of Eindhoven(TUE), Diciembre, 1993.

[16] D. Farmer y W. Venema, "Improving the Security of Your Site by Breaking Into it," *USENET newsgroup comp.security.unix*, ftp.win.tue.nl en /pub/security/admin-guide-to-cracking.Z, 1993.

[17] S. M. Bellovin y M. Merritt, "Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks," *Proceedings of*

the 1992 IEEE Computer Society Conference on Research in Security and Privacy, pp. 72 - 84, Mayo, 1992.

[18] M. Steiner, G. Tsudik, y M. Waidner, "Refinement and Extension of Encrypted Key Exchange," *ACM Operating Systems Review*, Vol.29, No.3, Julio, 1995.

[19] D. Jablon, "Strong Password-Only Authenticated Key Exchange," *Computer Communications Review*, Vol.26, No.5, pp. 5 - 26, Octubre, 1996.

[20] D. Jablon, "Extended Password Key Exchange Protocols Immune to Dictionary Attack," *WETICE '97 Enterprise Security Workshop*, Cambridge, MA, Junio, 1997.

[21] K.C. Goss, "Cryptographic Method and Apparatus for Public Key Exchange with Authentication," U.S.Patent 4956863, Septiembre, 1990.

[22] T. Matsumoto, Y. Takashima y H. Imai, "On Seeking Smart Public-Key-Distribution Systems," *The Transactions of the IECE of Japan*, Vol.E, No.69, pp. 99 - 106, 1986.

[23] M. Burrows, M. Abadi, y R.M. Needham, "A Logic for Authentication," *Technical Report 39, DEC System Research Center*, Palo Alto, California, Febrero, 1989. Versión Revisada de Febrero 22, 1990.

[24] M. Bellare y P. Rogaway, "Entity Authentication and Key Distribution," *Crypto '93 LNCS 773*, Springer-Verlag, pp. 232 - 249, Berlin, 1994.

[25] G. Tsudik y E,V. Herreweghen, "On Simple and Secure Key Distribution," *1993 ACM Conference on Computer and Communications Security*, Octubre, 1993.

[26] R.M. Needham y M.D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of ACM*, Vol.21, No.12, pp. 993 - 999, Diciembre, 1978.

[27] D. Denning y G. Sacco, "Timestamps in Key Distribution Systems," *Communications of ACM*, Agosto, 1981.

[28] R.L. Rivest, A. Shamir y L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of ACM*, Vol.21,No.2, pp. 120 - 126, Febrero 1978.

[29] T. ElGamal, "A Public Key Cryptosystem and Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, Vol.IT31,No.4, Julio 1985.

[30] D.R. Safford, D.K. Hess y D.L. Schales, "Secure RPC Authentication (SRA) for TELNET and FTP," *Proceedings of the Fourth USENIX Security Simposium*, 1993.

[31] National Bureau of Standards, "Encryption Algorithm for Computer Data Protection," *Federal Register*, Vol.40,No.52, pp. 12134 - 12139, Marzo 17, 1975.

[32] R.M. Needham y M.D. Schroeder, "Authentication Revisited," *ACM Operating Systems Review*, Vol.21, No.7, Enero, 1987, Vol.21, No.12, pp. 993 - 999, Diciembre, 1978.

[33] S. M. Bellovin y M. Merritt, "Limitations of the Kerberos Authentication System," *ACM SIGCOMM Computer Communication Review*, Octubre, 1990.

[34] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva y M. Yung, "A Modular Family of Secure Protocols for Authentication and Key Distribution," *IEEE/ACM Transactions on Networking*, Agosto, 1992.

[35] R. Molva, G. Tsudik, E.Van Herreweghen, S. Zatti, "KryptoKnight Authentication and Key Distribution Service," *Proceedings of ESORICS 92*, Toulouse, Francia, Octubre, 1992.

[36] G. Tsudik, E.Van Herreweghen, "Some Remarks on Protecting Weak Keys and Poorly-Chosen Secrets from Guessing Attacks," *1993 IEEE Symposium on Reliable Distributed Systems*, Octubre, 1993.

[37] L. Gong, "A Security Risk of Depending on Synchronized Clocks," *ACM Operating Systems Review*, Vo.26, No.1, Enero, 1992.

[38] B. Schneier, *Applied Cryptography Second Edition: protocols, algorithms, and source code in C*, John Wiley And Sons, Inc., 1996.

[39] W. Diffie y M.E. Hellman, "New Directions in Cryptography," *IEEE Transactions on Information Theory*, Vol. IT-11, pp. 644 - 654, Noviembre, 1976.

[40] S.C. Pohlig y M.E. Hellman, "An Improved Algorithm for Computing Logarithms over $GF(p)$ and its Cryptographic Significance," *IEEE Transactions on Information Theory*, pp. 106 - 110, Enero, 1978.

[41] S. Lucks, "Open Key Exchange: How to Defeat Dictionary Attacks Without Encrypting Public Keys," *Proceedings of the Security Protocol Workshop '97*, Springer-Verlag, Abril 7-9, 1997.

[42] N.M. Haller, "The S/KEY One-Time Password System," *Proceedings of the ISOC Symposium on Network and Distributed System Security*, Febrero 1994, San Diego, CA.

[43] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva y M. Yung, "Systematic Design of a Family of Attack-Resistant Authentication Protocols," *IEEE JSAC Special Issue on Secure Communication*, 1992, Springer-Verlag, 1991.

[44] L. Gong, "Criptographic Protocols for Distributed Systems," Tesis Doctoral, University of Cambridge, England, Abril 1990.

[45] W. Stallings, *Network and Internetwork Security Principles and Practice*, Prentice Hall, 1995.

[46] S. Goldwasser, S. Micali, y C. Rackoff, "The Knowledge Complexity of Interactive Proff Systems," *Proc. 17th Annual ACM Symp. Theory of Computing*, pp. 291 - 304, 1985.

[47] R. Oppliger, *Authentication Systems for Secure Networks*, Artech House Inc., 1996.

[48] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, Vol.21, Julio 1978, pp. 558 - 662.

[49] Ch. Kaufman, R. Perlman, M. Speciner, *Network Security PRIVATE Communication in a PUBLIC World*, Prentice Hall, Inc., 1995.

[50] A.J. Menezes, P.C. van Oorschot, S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, U.S.A., 1997.

[51] S. Garfinkel, G. Spafford *Web Security and Commerce*, OReilly and Associates, U.S.A., 1997.

[52] "Obsolete Password Methods", *http://world.std.com/~dpj/strong.html*

[53] J. Tardo, y K. Alagappan, "SPX: Global Authentication Using Public Key Certificates", *Proceedings of I.E.E.E. Computer Society Symposium on Research in Security and Privacy*, Oakland, pp. 232 - 244, Mayo 1991.

[54] L.D. Stein, "The World Wide Web Security FAQ (9. Client Side Security: Q58 How secure is the encryption used by SSL ?)", WWW Consortium, Abril, 1998. *http://www.w3.org/Security/faq/www-security-faq.html*

[55] L. Gong, "Optimal Authentication Protocols Resistant to Password Guessing Attacks", *Proceedings of the 8 IEEE Computer Security Foundations Workshop*, Country Kerry, Ireland, pp. 24 - 29, Junio 1995.

[56] S.M. Bellovin and M. Merrit, "An Attack on the Interlock Protocol when Used for Authentication", *IEEE Transactions on Informatyion Theory*, v.40, n.1, pp. 273 - 275, Enero 1994.

[57] C. Ellison, "Establishing Identy without Certification Authorities", *Proceedings of the Sixth Annual USENIX Securit y Symposium*, San Jose, pp. 67 - 76, Julio 1996.

[58] M. Roe, B. Christianson, D. Wheeler "Secure Sessions from Weak Secrets", *Technical Report from University of Cambridge and University of Hertfordshik*, Cambridge, 1998.

[59] W. Diffie, P.C. van Oorschot, and M. Winier "Authentication and Authenticated Key Exchanges", *Designs, Codes and Cryptography*, Vol. 2, pp. 107 - 125, 1992.

[60] S. Patel, "Number Theoretic Attacks On Secure Password Schemes ", *1997 IEEE Symposium on Security and Privacy*, Oakland California, Mayo 5-7, 1997.

[61] S. Halevi and H. Krawczyk, "Public-key Cryptography and Password Protocols", Por aparecer en *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, 1998.

[62] Steve Bellovin, Comunicación Privada, Agosto, 1998.

[63] Li Gong, Comunicación Privada, Agosto, 1998.

[64] Sarvar Patel, Commicación Privada, Agosto, 1998.

[65] R. J. Anderson and T.M.A. Lomas, "Fortifying Key Negotiation Schemes with Poorly Chosen Passwords", *Electronics Letters*, v. 30, no. 13, pp. 1040 - 1041, Junio 23, 1994.

[66] National Institute of Standards and Technology, NIST FIPS PUB 186, "Digital Signature Standard", U.S. Department of Commerce, Mayo, 1994.

[67] R. Lee and J. Israel, "Understanding the Role of Identification and Authentication in NetWare 4", *Novell Aplication Notes*, Octubre, 1994.

[68] R. Perlman and C. Kaufman, "Secure Password-Based Protocol for Downloading a Private Key", *Proceedings of the 1999 Netywork and Distributed System Security*, Febrero 3 - 5, 1999.

[69] T. Wu, "A Real-World Analysis of Kerberos Password Security", *Proceedings of the 1999 Netywork and Distributed System Security*, Febrero 3 - 5, 1999.

Apéndice A

Conceptos Básicos

A.1 Criptografía y Criptosistema

Informalmente hablando, la *criptografía* tiene como objetivo transformar la información para hacerla incomprensible a un lector no autorizado. Más formalmente, es el estudio de técnicas matemáticas relacionadas a aspectos de seguridad de la información tales como servicios de confidencialidad, integridad de los datos, y autenticación. Criptografía no sólo significa proporcionar seguridad a la información, sino más bien un conjunto de técnicas, llamadas técnicas criptográficas [50] (pp. 4).

Una forma de proporcionar servicios de seguridad es por medio de un *protocolo criptográfico*, el cual es un protocolo que utiliza técnicas criptográficas. Las técnicas que utiliza la criptografía son fundamentalmente tres: criptografía de llave secreta, simétrica o convencional; criptografía de llave pública o asimétrica, y funciones hash. Estas técnicas se revisan en las siguientes secciones de este apéndice. Un protocolo criptográfico es en este sentido, un algoritmo distribuido definido por una secuencia de pasos que especifican de manera precisa las acciones requeridas por dos o más entidades para alcanzar un objetivo de seguridad específico [50](pp. 33).

Toda técnica criptográfica toma como entrada la información que será transformada, la cual se conoce como *texto en claro* o *mensaje en claro* y lo

transforma, por medio de un *proceso de cifrado*, en un texto aparentemente aleatorio y sin ningún sentido que constituye la salida y es conocido como *texto cifrado* o *mensaje cifrado*. El proceso inverso, que transforma un texto cifrado en texto en claro, se llama *proceso de descifrado*.

El proceso de cifrado consta de un *algoritmo* y una *llave*. La llave es un valor independiente del texto en claro y controla el algoritmo. El algoritmo producirá salidas distintas, dependiendo de la llave usada.

Un *criptosistema* es un esquema de cifrado y descifrado que contiene, como mínimo, los siguientes elementos: el *generador* o *fuente* del texto en claro o mensaje en claro X; el algoritmo de cifrado E, que aplicado a X con la llave K, produce el texto cifrado Y; el algoritmo de descifrado D que aplicado a Y con la llave K produce el texto en claro X, que es entregado al *destino* del mensaje.

A.2 Técnicas Criptográficas

La criptografía se clasifica, de acuerdo al número de llaves que involucra en las siguientes técnicas: *de llave secreta, simétrica o convencional, de llave pública o asimétrica* y *funciones hash*.

A.2.1 Criptografía de Llave Secreta, Simétrica o Convencional

La criptografía de llave secreta, simétrica o convencional utiliza una única llave, llamada *llave secreta*, tanto para cifrar como para descifrar, manteniendo simetría con respecto a los procesos de cifrado y descifrado. Este es el tipo de criptografía que clasicamente se ha utilizado, por eso se conoce también como convencional. Esta técnica criptográfica presupone que tanto el origen como el destino comparten previamente un secreto o llave secreta, que de alguna manera, se ha distribuido con anterioridad.

Algunos de los algoritmos de llave secreta más conocidos en la actualidad son el *DES ("Data Encryption Standard")* [50] (pp. 250) y el *IDEA ("International Data Encryption Algorithm")* [50] (pp. 263).

A.2.2 Criptografía de Llave Pública o Asimétrica

La criptografía de llave pública o asimétrica utiliza dos llaves distintas pero relacionadas entre sí que se generan por parejas: la *llave pública*, que es públicamente expuesta para que cualquiera la pueda consultar y utilizar, y la *llave privada* que debe mantenerse en secreto. El punto aquí es que lo que se cifra con una de las llaves se tiene que descifrar con la otra que le corresponde. En este caso, la simetría no se mantiene, motivo por el cual también se conoce a esta técnica como asimétrica. En este caso no se necesita intercambiar previamente un secreto que sirva como llave de cifrado. Por ejemplo, si el servicio que se desea es confidencialidad, la llave para cifrar se consulta de manera pública.

El algoritmo de llave pública más conocido y utilizado en la actualidad es el *RSA ("Rivest, Shamir, Adleman")* [50] (pp. 285).

A.2.3 Funciones Hash

Esta técnica criptográfica no utiliza ninguna llave. Una *función hash* también se conoce como *función de compresión*, *dispersión*, *contracción* o *compendio de mensaje*. Es una función que toma como entrada una cadena de longitud variable, llamada *preimágen*, y la convierte en una cadena de salida de longitud fija, generalmente más pequeña alrededor de 128 bits), llamada *resultado o valor hash*.

Una función *hash unidireccional* es una función hash que es fácil de calcular en una dirección, pero computacionalmente infactible de invertir. Si $y = f(x)$, donde f es la función hash unidireccional con entrada x y salida y, entonces calcular y dado x es fácil y rápido; pero hallar x' tal que $y = f(x')$ para una y dada, es extremadamente difícil. Idealmente, no debe haber ningún otro modo de determinar x' que no sea intentando un número infactible de valores y ver cuál de ellos es correcto. Para todo propósito práctico, la función no puede ser invertida [50] (pp. 321).

En el contexto de este trabajo, cuando se hable de funciones hash, se entenderá que se habla de funciones hash unidireccionales. Las principales funciones hash conocidas y ampliamente utilizadas son: $MD4$, $MD5$, $MD2$

[38] (pp. 435 - 441). Si se desea profundizar en el conocimiento sobre las funciones hash, pueden consultarse [38, 49, 50].

A.3 Criptoanálisis

Al proceso que realiza un atacante con el fin de averiguar el texto en claro, la llave de cifrado, o ambos, se le llama *criptoanálisis*. La estrategia de ataque usada por el *criptoanalista* depende de la naturaleza del esquema de cifrado y de la información de que dispone el atacante.

Los ataques criptográficos pueden dirigirse contra los algoritmos criptográficos, contra las técnicas criptográficas usadas para implementar los algoritmos y protocolos, y contra los propios protocolos. Cuando se estudian protocolos criptográficos, normalmente se asume que los algoritmos criptográficos y las técnicas de implementación son seguros. Más detalle sobre este tema puede hallarse en [38, 49, 50].

De acuerdo con lo anterior, en el presente trabajo no se analizan ataques por criptoanálisis porque se asume que los algoritmos criptográficos y sus implementaciones son seguros. No obstante, dada su importancia y relación con los protocolos criptográficos, se mencionan a continuación los principales ataques por criptoanálisis.

A.3.1 Ataques por Criptoanálisis

Los principales ataques a un criptosistema, usando criptoanálisis, son los siguientes [38] (pp. 5 - 7) y dependen de la cantidad y tipo de información que el criptoanalista posee. En todos los casos se asume que el criptoanalista conoce el algoritmo de cifrado:

1. *Sólo texto cifrado.*- El atacante sólo conoce el texto cifrado de varios mensajes cifrados con el mismo algoritmo.
2. *Texto en claro conocido.*- El atacante conoce el texto cifrado de varios mensajes y una o más parejas (texto en claro, texto cifrado).

3. *Texto en claro escogido.*- El atacante no sólo conoce el texto cifrado y su correspondiente texto en claro para varios mensajes, sino que puede escoger el texto en claro que desea conocer cifrado.

4. *Texto en claro escogido adaptivo.*- Este es un caso especial de *texto en claro escogido.* El atacante no sólo escoge el texto en claro que se cifrará, sino que también puede modificar su elección basándose en resultados de cifrados previos.

5. *Texto cifrado escogido.*- El atacante puede escoger diferentes textos cifrados y conocer los correspondientes textos en claro para varios mensajes.

6. *Texto escogido.*- Este caso junta un ataque *texto en claro escogido* y un *texto cifrado escogido.* El atacante no sólo conoce el texto cifrado y su correspondiente texto en claro para varios mensajes, sino también puede escoger el texto en claro que desea conocer cifrado; no sólo conoce el texto en claro y su correspondiente texto cifrado para varios mensajes, sino también puede escoger el texto cifrado y conocer el correspondiente texto en claro.

A.4 Suposiciones Criptográficas usadas en Autenticación

Se dice que una llave de cifrado es criptográficamente "mala o débil" si es derivada de un password seleccionado por el usuario, o el password mismo es criptográficamente débil si se usa como llave. Se considera criptográficamente "buena o fuerte" una llave de cifrado si se selecciona de manera aleatoria, de un espacio de llaves grande. Esta distinción se basa en la presumiblemente baja probabilidad de adivinar o conjeturar exitosamente, por parte de un atacante, una llave aleatoriamente seleccionada, comparada con la presumiblemente alta probabilidad de adivinar o conjeturar exitosamente un password seleccionado por el usuario [1].

Las soluciones al problema de autenticación que utilizan técnicas criptográficas se basan en algoritmos criptográficos ampliamente conocidos y probadamente robustos, tanto de criptografía convencional como de llave pública. Es por esta razón que se asume que los criptosistemas no son vulnerables con

respecto a la confidencialidad e integridad de los mensajes intercambiados, porque estos servicios los proporcionan los algoritmos criptográficos usados. De esta manera, al estudiar protocolos criptográficos no se consideran ataques por criptoanálisis.

A.5 Firmas Digitales

Informalmente, una *firma digital* es el *mecanismo de autenticación* que permite al creador de un mensaje, anexarle a ese mensaje un código que actúa como una firma manuscrita. La firma garantiza el origen y la integridad del mensaje [45] (pp. 444).

Más formalmente, es una función primitiva criptográfica que proporciona autenticación, autorización y no repudio. Proporciona un medio para que una entidad relacione su identidad a una pieza de información conocida como documento o simplemente mensaje. La firma depende de la identidad del firmante a través de un secreto o llave, y del documento mismo. Consta de dos procesos fundamentales: la firma y la verificación de la firma. La firma es una transformación que vincula el documento y el secreto mantenido por la entidad; la verificación es una transformación que va del producto cartesiano de mensajes y firmas, a un conjunto de dos posibles valores: verdadero y falso. Si la verificación es verdadera, se acepta la firma como creada por el firmante; si es falsa, se rechaza [50] (pp. 22 - 23).

Firmar el mensaje M con la llave privada K^r, se expresa como $S_{K^r}(M)$, y verificar la firma X del mensaje M con llave pública K^p, se expresa abreviadamente como $V_{K^p}(X, M)$. La cadena de bits agregada al documento cuando se firma (por ejemplo, la función hash del documento, cifrada con la llave privada) es lo que constituye la *firma digital* o simplemente *firma* [38] (pp. 39).

Existen muchos algoritmos para firmas digitales. Todos ellos son algoritmos de llave pública con información secreta para firmar documentos y con información pública para verificar las firmas. Entre los algoritmos ampliamente conocidos que proporcionan cifrado y firma digital están: *RSA* [38] (pp. 466 - 474), ElGamal [38] (pp. 477 - 478), y Rabin [38] (pp. 475 - 476).

Algunas veces el proceso de firma se llama *cifrado con llave privada* y el proceso de verificación se conoce como *descifrado con llave pública*. Esto es engañoso y es cierto sólo para el *RSA*. Diferentes algoritmos tienen distintas implementaciones. Muchos algoritmos pueden usarse para firmas digitales, pero no para cifrado. En general, se hace referencia a los procesos de firma y verificación sin ningún detalle sobre el algoritmo involucrado.

Para profundizar sobre este tema, pueden consultarse [38, 49, 50].

A.6 S/KEY

S/KEY [42] es un esquema de autenticación que basa su seguridad en una función hash unidireccional. La explicación informal de su funcionamiento es la siguiente [38] (pp. 53): A genera un número aleatorio R y calcula $x_1 = f(R)$, $x_2 = f(f(R))$, $x_3 = f(f(f(R)))$, y así hasta cien veces. A imprime esta lista de números y lo guarda en un lugar seguro. También calcula el valor x_{101} y lo guarda en la base de datos de *"logins"* después de la identidad de A.

La primera vez que A desea acceso (*"login"*), proporciona su identidad y el valor x_{100} a B. B calcula $f(x_{100})$ y lo compara con x_{101}; si coinciden, A es autenticado y obtiene acceso. Si es así, B reemplaza x_{101} con x_{100} en la base de datos de *"logins"*. A borra x_{100} de su lista.

Cada vez que A accesa, proporciona a B el último número no borrado de su lista, x_i. B calcula $f(x_i)$ y compara el resultado con el x_{i+1} guardado en su base de datos.

Para más información sobre este esquema, basado en el protocolo de Lamport, puede consultarse [50] (pp. 396).

A.7 SSL (*"Secure Socket Layer"*)

El *SSL* es un protocolo criptográfico de propósito general, usado en la actualidad para asegurar canales de comunicación bidireccionales. *SSL* se

utiliza comúnmente con el protocolo de *Internet, TCP/IP*. Es el sistema de seguridad que usan los visualizadores o navegadores, tales como *Netscape Navigator* y *Internet Explorer* de *Microsoft*, pero puede usarse con cualquier servicio *TCP/IP*.

Las conexiones *SSL* se inician usualmente con un visualizador, por medio del uso de un prefijo especial *URL*. Por ejemplo, el prefijo *"https:"* se usa para indicar una conexión *HTTP* cifrada con *SSL*, mientras *"snews:"* se usa para indicar una conexión *NNTP* cifrada con *SSL*.

SSL proporciona confidencialidad usando algoritmos de cifrado especificados por el usuario; ofrece integridad por medio del uso de funciones hash especificadas por el usuario; proporciona autenticación, a través del uso de certificados de llave pública estandarizados por *X.509 v3*; y ofrece no repudio a través de mensajes firmados criptográficamente.

Para más información sobre este protocolo, puede consultarse [51].

A.8 Pruebas de Conocimiento Cero

Una prueba de *conocimiento cero* permite demostrar que se conoce un secreto sin revelar ese secreto a nadie. Conocido en la literatura como *ZK* (*"Zero Knowledge"*) o *ZKIP* (*"Zero-Knowledge Interative Proofs"*), ha recibido considerable atención desde 1984 a partir del trabajo de Goldwasser, Micali y Rackoff [46] en el cual examinan la cantidad de conocimiento que debe ser communicado para convencer a un verificador, en tiempo polinomial, acerca de la validez de un teorema.

Las técnicas *ZKIP* son útiles en muchas áreas de aplicación, tales como identificación y firmas digitales en sistemas de autenticación.

Para mayor información sobre este tema, pueden consultarse las referencias [38, 49, 50].

A.9 Entropía

Sea X una variable aleatoria que toma un conjunto finito de posibles valores x_1, x_2,..., x_n, con probabilidad $P(X = x_i) = p_i$, donde $0 \leq p_i \leq 1$ para toda i, $1 \leq i \leq n$, y donde

$$\sum_{i=1}^{n} p_i = 1 \tag{A.1}$$

La *entropía* de X, donde X es una variable aletaoria que toma un conjunto finito de valores, es una medida de la cantidad de información proporcionada por una observación de X. Equivalentemente, es la *incertidumbre* acerca del resultado antes de una observación de X. La entropía es también útil para aproximar el número promedio de bits requeridos para codificar los elementos de X [50] (pp. 56).

La entropía o incertidumbre de X se define como

$$H(X) = -\sum_{i=1}^{n} p_i(logp_i) = \sum_{i=1}^{n} p_i log(\frac{1}{p_i}) \tag{A.2}$$

donde por convención p_i $(log\ p_i) = p_i\ log\ (\frac{1}{p_i}) = 0$ si $p_i = 0$ [50] (pp. 56).

A.10 Grupos

Un *grupo* $(G, *)$ consiste de un *conjunto* G con una operación binaria * sobre G, que satisface las siguientes tres condiciones:

1. La operación es *asociativa*. Es decir, $a * (b * c) = (a * b) * c$, $\forall\ a, b, c \in G$.
2. $\exists\ 1 \in G$, llamado *elemento identidad*, tal que $a * 1 = 1 * a = a$, $\forall a \in G$.
3. $\forall\ a \in G$, $\exists\ a^{-1} \in G$, llamado el *inverso* de a, tal que $a * a^{-1} = a^{-1} * a = 1$.
4. El grupo G es *abeliano* o *conmutativo* si, además, $a * b = b * a$, $\forall\ a, b \in G$.

Es de notarse que en la definición anterior se ha usado el operador multiplicativo * para la operación del grupo. Si la operación es de suma, entonces el grupo se dice que es *aditivo*, el elemento identidad se denota por 0 y el inverso de a se denota por $-a$.

Un grupo G es finito si $|G|$ es finito. El número de elementos en un grupo finito es llamdo su *orden*.

A.10.1 Z_n, Z_n^*, y Cociente de Euler

El conjunto de enteros *módulo* n, denotado por Z_n, es el conjunto de enteros $\{0, 1, 2, ..., n-1\}$.

El grupo multiplicativo de Z_n es $Z_n^* = \{a \in Z_n, |mcd(a, n) = 1\}$, donde $mcd(a, n)$ es el *máximo común divisor* de a y n. En particular, si n es primo, entonces $Z_n^* = \{a, |1 \leq a \leq n-1\}$.

Para $n \geq 1$, $\varphi(n)$ denota el número de enteros en el intervalo $[1, n]$ que son primos relativos con n. La función φ es conocida como *función φ de Euler* o *función cociente de Euler*. Una de sus propiedades importantes es que, si p es primo, entonces $\varphi(p) = p$ - 1.

De la definición de φ se sigue que el orden de Z_p^*, es $\varphi(p)$.

A.10.2 Subgrupos

Un subconjunto no vacío H de un grupo G es un *subgrupo* de G si H mismo es un grupo con respecto a la operación de G. Si H es un subgrupo de G y H es distinto de G, entonces H es llamado un em subgrupo propio de G.

Un grupo G se dice que es *cíclico* si $\exists\ \alpha \in G$ tal que $\forall\ b \in G$, $\exists$ un entero i con $b = \alpha^i$. Tal elemento es llamado un *generador* de G.

Si G es un grupo y $a \in G$, entonces el conjunto de todas las potencias de a forman un subgrupo cíclico de G, llamado el subgrupo generado por a, y denotado por $\langle a \rangle$.

$\forall x$ que es divisor de p - 1, el grupo Z_p^* contiene un subgrupo G_x de orden x, es decir, G_x contiene x elementos. De esta manera, $Z_p^* = G_{p-1}$. Cada grupo G_x contiene el *subgrupo trivial* G_1, consistente de $\{1\}$ y el subgrupo G_x mismo.

Un grupo G_q de orden q, donde q es primo, no tiene ningunos otros subgrupos que G_q y G_1. Un *generador* g de G_x es un número tal que el conjunto $\{g^1, g^2, ..., g^x\}$ incluye todos los elementos de G_x. Un generador de G_{p-1} se dice que es una *raíz primitiva* de p. Un número se dice que es de *orden* x cuando es un generador de G_x. Si g es de orden x, entonces $g^x \equiv 1 \bmod p$.

A.11 Acuerdo de Llave Diffie-Hellman

El *acuerdo, establecimiento o intercambio de llave Diffie-Hellman*, también conocido como *intercambio de llave exponencial* es una técnica fundamental que proporciona lo que se conoce como acuerdo de llave autenticada (Sección 3.4 del Capítulo 3 de este trabajo).

El acuerdo de llave Diffie-Hellman proporciona la primera solución práctica al problema de distribución de llave, al permitir a dos partes, sin acuerdo previo ni compartiendo información alguna sobre llaves, establecer un secreto compartido a través del intercambio de mensajes sobre un canal abierto. Su seguridad descansa sobre la intratabilidad del *Problema Diffie-Hellman* y el problema relacionado de calcular *logaritmos discretos*. Ambos problemas se enuncian a continuación.

A.11.1 Problema Diffie-Hellman

El Problema Diffie-Hellman es el siguiente: dado un número primo p, un generador α de Z_p^*, y los elementos $\alpha^a \ mod \ p$ y $\alpha^b \ mod \ p$, hallar $\alpha^{ab} \ mod \ p$.

La versión *generalizada* del Problema Diffie-Hellman es la siguiente: dado un grupo cíclico finito G, un generador α de G, y los elementos del grupo α^a y α^b, hallar α^{ab}.

A.11.2 Problema del Logaritmo Discreto

Sea un grupo cíclico finito G de orden n, un generador α de G, y $\beta \in G$. El *logaritmo discreto de β a la base α*, denotado por $log_\alpha\ \beta$, es el único entero x, $0 \leq x \leq n-1$, tal que $\beta = \alpha^x$.

El *problema del logaritmo discreto* es el siguiente: dado un número primo p, un generador α de Z_p^*, y $\beta \in Z_p^*$, hallar el entero x, $0 \leq x \leq p-2$, tal que $\alpha^x \equiv \beta \bmod p$.

La versión *generalizada* del Problema del Logaritmo Discreto es la siguiente: dado un grupo cíclico finito G de orden n, un generador α de G, y $\beta \in G$, hallar el entero x, $0 \leq x \leq n-1$, tal que $\alpha^x = \beta$.

A.11.3 Protocolo Básico Diffie-Hellman

La versión básica del acuerdo de llave Diffie-Hellman proporciona confidencialidad de la llave resultante contra atacantes pasivos, pero no contra atacantes activos capaces de interceptar, modificar e insertar mensajes. Ninguna de las partes tiene la certeza de la identidad de la fuente u origen del mensaje de entrada o de la identidad de la parte que puede conocer la llave resultante. El protocolo es el siguiente:

Previo a la ejecución del protocolo, las partes seleccionan y publican un número primo p adecuado [1] y un generador α de Z_p^*, $0 \leq \alpha \leq p-2$.

1. A selecciona aleatoriamente un número x, $1 \leq x \leq p-2$, calcula $\alpha^x \bmod p$ y lo envía a B:

   $A \longrightarrow B : \alpha^x \bmod p$

2. Al recibir $\alpha^x \bmod p$, B selecciona aleatoriamente un número y, $1 \leq y \leq p-2$. Calcula la llave de sesión $k = (\alpha^x)^y \bmod p$. Calcula $\alpha^y \bmod p$ y lo envía a A:

   $B \longrightarrow A : \alpha^y \bmod p$

[1] El tamaño considerado como adecuado en términos de seguridad para p es de 1024 bits [19] (pp. 17)

Apéndice B

Información sobre el Tema y Areas Relacionadas

B.1 Información sobre Protocolos Fuertes

Para mantenerse actualizado sobre el estado, los avances y la investigación sobre protocolos fuertes de autenticación basada en passwords y áreas relacionadas, existen algunas direcciones en el Web donde se pueden conseguir artículos recientes e información genérica sobre el área.

1. *http://world.std.com/~dpj/*
2. *http://www.num.math.uni-goettingen.de/lucks/papers.html*

B.2 Información sobre Eventos y Referencias Varias

Algunas direcciones de interés por su información sobre realización de eventos tales como conferencias, cursos, seminarios, y por sus referencias sobre libros, notas, periódicos y revistas sobre seguridad y criptografía en general, son las siguientes:

1. *http://www-cse.ucsd.edu/users/mihir/crypto-links.html*
2. *http://www.enter.net/~chronos/cryptolog1.html*

B.3 Información de Revistas Periódicas sobre el Tema

Algunas de las principales Revistas periódicas que publican artículos sobre criptografía y temas afines son las siguientes:

1. *Communications of the ACM*
2. *Computer Communications Review*
3. *CryptoBytes*
4. *Designs, Codes and Cryptography*
5. *Electronics Letters*
6. *IEEE Journal on Selected Areas in Communications*
7. *IEEE Transactions on Information Theory*
8. *Journal of Cryptology*
9. *Operating Systems Review*

B.4 Simposios, Conferencias y *"Workshops"*

Algunos de los principales Simposios, Conferencias, y *"Workshops"* que se realizan anualmente, principalmente en territorio de los Estados Unidos de Norteamérica, sobre temas actuales de Seguridad, Confidencialidad y Comunicaciones, entre otros, son los siguientes:

1. *ACM Conference on Computer and Communications Security*

2. *ACM Symposium on Operating Systems Principles*
3. *ACM Symposium on the Theory of Computation (STOC)*
4. *Annual USENIX Security Symposium*
5. *Annual USENIX Security Conference*
6. *IEEE Computer Security Foundations Workshop*
7. *IEEE Computer Society Symposium on Research in Security and Privacy*
8. *IEEE Symposium on Research in Security and Privacy*
9. *Internet Society Symposium on Network and Distributed System Security*
10. *The Conference on Computer Communications*

B.5 *"Proceedings"* de Conferencias, Simposios y *"Workshops"*

La International Association for Cryptologic Research (IACR) se dedica principalmente a la organización de reuniones internacionales. Los artículos presentados en estas conferencias se publican en la serie Lecture Notes in Computer Science (LCNS) de la editorial Springer Verlag.

La siguiente es la lista de los "proceedings" de 1990 a 1997 de la reunión anual en la Universidad de Santa Bárbara, California:

1. ***Advances** in Cryptology: CRYPTO'90*, A.J. Menezes, S.A. Vanstone Eds., **LCNS 537**, 1991.
2. ***Advances** in Cryptology: CRYPTO'91*, J. Feigenbaum Ed., **LCNS 576**, 1992.
3. ***Advances** in Cryptology: CRYPTO'92*, E.F. Brickell Ed., **LCNS 740**, 1993.

4. *Advances in Cryptology: CRYPTO'93*, D.R. Stinson Ed., **LCNS 773**, 1994.

5. *Advances in Cryptology: CRYPTO'94*, Y.G. Desmedt Ed., **LCNS 839**, 1994.

6. *Advances in Cryptology: CRYPTO'95*, D. Coppersmith Ed., **LCNS 963**, 1995.

7. *Advances in Cryptology: CRYPTO'96*, N. Koblitz Ed., **LCNS 1109**, 1996.

8. *Advances in Cryptology: CRYPTO'97*, B.S. Kaliski Ed., **LCNS 1294**, 1997.

La siguiente es la lista de los *"proceedings"* de 1990 a 1997 de la reunión anual conocida como *bf EUROCRYPT*, que se lleva a cabo en una ciudad diferente de Europa:

1. *Advances in Cryptology: EUROCRYPT'90*, I.B. Damgard Ed., **LCNS 473**, Aarhus Denmark, 1991.

2. *Advances in Cryptology: EUROCRYPT'91*, D.W. Davies Ed., **LCNS 547**, Brighton United Kingdom, 1991.

3. *Advances in Cryptology: EUROCRYPT'92*, R.A. Rueppel Ed., **LCNS 658**, Balatonfured Hungaria, 1993.

4. *Advances in Cryptology: EUROCRYPT'93*, T. Helleseth Ed., **LCNS 765**, Lofthus Norway, 1994.

5. *Advances in Cryptology: EUROCRYPT'94*, A. DeSantis Ed., **LCNS 950**, Perugia, Italy, 1995.

6. *Advances in Cryptology: EUROCRYPT'95*, L.C. Guillou, J.J. Quisquater Eds., **LCNS 921**, Saint-Malo France, 1995.

7. *Advances in Cryptology: EUROCRYPT'96*, U. Maurer Ed., **LCNS 1070**, Saragossa Spain, 1996.

8. *Advances in Cryptology: EUROCRYPT'97*, W. Fumy Ed., **LCNS 1233**, Konstanz Germany, 1997.