UNIVERSIDAD DE PANAMÁ

VICERRECTORÍA DE INVESTIGACIÓN Y POSTGRADO

PROGRAMA DE MAESTRÍA EN INGENIERÍA DE SISTEMAS DE COMUNICACIONES CON ÉNFASIS EN REDES DE DATOS

ANÁLISIS Y PROPUESTA DE UN PROTOCOLO DE AUTENTICACIÓN ROBUSTA MEDIANTE ONE-TIME PASSWORDS USANDO TARJETAS INTELIGENTES PARA REDES DIGITALES EN EL HOGAR.

SANDOR ERNESTO TUÑÓN ANDRÉS

TESIS PRESENTADA COMO UNO DE LOS REQUISITOS PARA OBTENER EL GRADO DE MASTER EN INGENIERÍA DE COMUNICACIONES CON ÉNFASIS EN REDES DE DATOS

PANAMÁ, REPÚBLICA DE PANAMÁ

2012

Hoja de aprobación

DEDICATORIA

A mi esposa y mi madre

AGRADECIMIENTOS

A mi madre por haber hecho de mí el hombre que soy, por estar siempre ahí y por su amor incondicional.

A mi esposa por su amor, su valiosa ayuda y su paciencia sin límites.

A mi familia y amigos, quienes me han apoyado espiritual y materialmente para llegar a este momento.

Al SENACYT y la Universidad de Panamá por permitirme la magnífica oportunidad de cursar esta maestría y ampliar así mis aéreas de conocimiento.

Al Dr. Javier García Villalba, Co-Director de la tesis, Director del GASS (Grupo de Análisis Seguridad y Sistemas) de la Universidad Complutense de Madrid; quien me ha guiado a lo largo de todo mi proyecto; recibiéndome, además, durante dos meses de intensa labor en el grupo, cuya amable colaboración me permitió llevar a un estado maduro este trabajo.

A los colegas del GASS, que me permitieron una integración magnífica en su grupo y participaron con ánimo constructivo en proceso de análisis y discusiones progresivas de la presente investigación, impulsando el avance de la misma con sus valiosos comentarios y sugerencias.

Al Dr. Ivan Armuelles por apoyo y disposición constante de ayudar en todo cuanto le es posible; por darme la oportunidad de trabajar y compartir con él y estar así siempre que lo necesitamos para darnos la mano, para animarnos, para impulsarnos. Cabe señalar que sin él las condiciones en que se desarrollado la última parte de este trabajo hubieran sido indudablemente más difíciles.

Al los estimados profesores Gustavo Díaz y Álvaro Maturel por su constante cooperación y comprensión, así como del resto de los profesores e investigadores que trabajaron con nosotros como parte del programa.

A la señora Jane Saldaña por presionarnos para realizar investigaciones meritorias.

A todos aquellos de de una forma u otra han me han apoyado y han facilitado directa o indirectamente la realización de este trabajo de Tesis.

TABLA DE CONTENIDO

ÍNDICE DE CUADROS

ÍNDICE DE FIGURAS

GLOSARIO DE ABREVIATURAS

AAA: Autentication, Autorization and Acconting

DoS: Denial of Service attack

HG: Home Gateway

HMAC: Hash Message Authentication Code

HOTP: HMAC based One Time Password algorithm

IAS: Integrated Authentication Server

IETF: Internet Engineering Task Force

IPsec: Internet Protocol security

ISO: International Organization for Standardization

MAC: Message Authentication Code

MD5: Message Digest version 5

MDC: Modification Detection Code

nonce: number used once

OATH: initiative for Open AuTHentication

OSPA: Optimal Strong-Password Authentication

OTP: One Time Password

PIN: Personal Identification Number

PW: PassWord

RFC: Request For Comments

RSA: Rivest, Shamir y Adleman

SAS: Simple And Secure password authentication protocol

SC: Smart Card

SHA1: Secure Hash Algorithm version 1

TTP: Trusted Third Party

U: User

WEP: Wired Equivalent Privacy

XOR: eXclusive OR

GLOSARIO DE SÍMBOLOS

Símbolo	Descripción
$Im(f)$	Imagen de f
$f: X \to Y$	Función definida en un domino X con imagen Y
$\Rightarrow$	Implica
$\|C\|$	Cardinal del conjunto C
$\gg$	Mucho mayor que
$\in$	Pertenece
$\forall$	Para todo elemento
$\oplus$	Operación XOR
$\|\|$	Operación de concatenación
$H^n(S)$	Función hash H anidada, aplicada n-veces a S
$H^i(K, C)$	*i*-ésimo valor de HOTP con los parámetros K y C
C_X	Contador de 8-bytes, factor de cambio (moving factor) X = C – cliente, S – servidor, M – máximo permitido
K	Secreto compartido entre el Usuario y el Servidor
K_i, S_K	Claves de sesión
K_{AB}	Clave secreta compartida entre A y B
K_{IAS-HG}	Clave secreta preestablecida entre IAS y HG
T	Trudy (atacante)
N	Número de accesos permitidos
S	Semilla aleatoria
R	Valor aleatorio (Random)
N_A , N_B	Nonce de A, nonce de B
R_A, R_B	Aleatorio generado por A, aleatorio generado por B
T_A, T_B	Marca de tiempo de A, marca de tiempo de B (time-stamp)

S_n	*n*-ésimo valor de una secuencia
$E_{K_X}(M)$	Cifrado simétrico con la clave K_X
TKG	Tiquete brindado para la autenticación (TicKet Granted)
T_{exp}	Tiempo de expiración para el tiquete de autenticación
$A \longrightarrow B$	Envío de información de A hacia B
$A \Longrightarrow B$	Envío de información de A hacia B usando un canal seguro
PW_N	Nueva contraseña
M'	Usado para identificar un mensaje que supuestamente es M, pero en realidad podría diferir de M
$\mathcal{E}(M,k),\ \mathcal{E}_k\{M\}$	Cifrado del valor M con clave K
$V \overset{?}{=} V'$	Verificación de la igualdad de dos valores

.

RESUMEN

En este trabajo, se realiza un análisis de un protocolo de autenticación robusta usando tarjetas inteligentes para *redes digitales en el hogar*, propuesto por Vaidya, Hyuk, Yeo y Rodríguez (2011). Este tipo de redes digitales ha ido creciendo en aplicaciones para el confort y la atención médica a distancia, entre otras opciones. En estos entornos delicados, no siempre se puede garantizar la seguridad de los canales de comunicación, por lo que se requiere de otras alternativas que brinden un alto nivel de seguridad. Los actuales mecanismos de autenticación robusta usando OTPs (*One Time Passwords*) vienen a ser una solución factible para ello. Teniendo en cuenta que normalmente en este tipo de redes intervienen clientes inalámbricos, se han venido desarrollando una serie de protocolos basándose en operaciones criptográficas ligeras, como es el caso del protocolo analizado en este trabajo. Estas operaciones usadas son preferentemente las operaciones *Hash* y *XOR*. Las *cadenas de hash* han sido el mecanismo preferido en este tipo de esquemas, sin embargo, suelen implicar un alto número de operaciones para, al menos, una de las partes involucradas, lo cual atenta contra la eficiencia computacional. Teniendo en cuenta que los protocolos de seguridad son bastante sutiles, cualquier protocolo planteado debe ser sometido a una extensa revisión y análisis en busca de deficiencias y vulnerabilidades. En la presente investigación no sólo se analizan algunas vulnerabilidades del protocolo de Vaidya et al. (2011), sino que también se proponen mejoras a la seguridad y la eficiencia de este protocolo, optimizando de este modo su funcionamiento.

PALABRAS CLAVES: *protocolo de autenticación robusta; cadenas de hash; tarjetas inteligentes; OTP; one time password; redes digitales en el hogar*

ABSTRACT

In this thesis we discuss about the security and efficiency of a *robust authentication scheme* using *smart cards* for accessing *digital home networks*, proposed by Vaidya, Hyuk, Yeo and Rodríguez (2011). Digital home networks have been employed in recent times in applications increasing comfort, as well as remote healthcare and more others. In these delicate environments, communication channel protection cannot always be easy to guarantee. So other alternatives that offer a high level of security are required. The actual mechanisms of robust authentication, using OTPs *(one time passwords)*, are a feasible solution to this concerning. Taking into account that generally in this kind of networks wireless clients are usually involved, this led to the developing a series of protocols based in light cryptographic operations, as it's the case of the protocol analyzed in this thesis. These operations are preferably *Hash* and *XOR*. The *hash chains* has been the preferred mechanism in this types of schemes; however it used to involve a high amount of operations for, at least, one of the parts implicated, so it attempt against computing efficiency. Based on the fact that the security protocols are quite sensitive, any protocol proposed should be subjected to an extensive review looking for vulnerabilities. In the present research, not only we analyze some vulnerabilities of the Vaiyad et al. (2011) protocol, but we also propose improvements in order to solve some of the security vulnerabilities and enhance the computational efficiency of the protocol.

KEYWORDS: *Robust authentication scheme; hash chain; smart card; one time passwords; OTP; digital home networks.*

INTRODUCCION

El desarrollo que se ha alcanzado en la protección de la información hoy en día, no hubiera sido posible sin el uso indispensable de la *Criptografía*. En la actualidad se han logrado enormes avances en esta rama de las ciencias, haciendo uso de bases matemáticas y los enormes avances tecnológicos existentes. Así, el acelerado avance de las tecnologías ha permitido diversas estrategias para la protección de la información y la seguridad informática, pero al mismo tiempo ha traído consigo nuevas amenazas. Aunque en la actualidad la información se puede proteger mucho mejor de aquellos que carecen de conocimientos suficientes, o no disponen de los recursos apropiados para romper los métodos de protección, debemos darnos cuenta de que la tecnología existe y se desarrolla tanto para aquellos que necesitan proteger la información, como para aquellos que pretenden romper los mecanismos de seguridad existentes. De este modo, podríamos decir que este tema de la seguridad informática ha sido, y será siempre, una lucha incesable entre aquellos que trabajan en función de crear nuevos mecanismos y estrategias para la protección de la información, y aquellos que se esfuerzan por buscar sus vulnerabilidades y tratan de violar los mismos, con el respaldo de una tecnología equivalente.

Debido al crecimiento exponencial de los usuarios de Internet, y los dispositivos inalámbricos conectados en todo momento a la gran red de redes, la computación ubicua ha tomado un papel importante en el progreso de las tecnologías y las redes de hoy día. Las redes domésticas avanzadas permiten conectividad a los usuarios del hogar a través de Internet, para el acceso y el control de dispositivos digitales interconectados entre sí y asociados a la vida en el hogar.

Las *redes digitales en el hogar* son una de las tecnologías representativas del fenómeno de la computación ubicua. Este tipo de redes se dedican fundamentalmente al confort, ocio y cuidado de la salud de los moradores del hogar; aunque también han sido validadas para proteger lugares y para optimizar procesos como el consumo de energía.

Aún cuando las redes digitales proveen conveniencias a los usuarios residenciales, brindando servicios de valor agregado; las características heterogéneas de las mismas, y lo delicado del ambiente, hacen que resulte un desafío importante la protección de la privacidad y confidencialidad de la información que se transmite. En la mayoría de los casos es deseable que los usuarios legítimos puedan realizar el acceso remoto a distintos servicios que estas redes brindan. Sin embargo, a no ser que estas redes estén bien protegidas, usuarios ilegítimos podrían conseguir acceso a estos servicios, poniendo en riesgo la seguridad del hogar. Así, la autenticación de usuarios es uno de los mecanismos de seguridad más importantes requeridos en las redes digitales para el hogar, el cual ha generado creciente interés de la comunidad científica.

Los mecanismos de autenticación robusta actualmente existentes, en particular los mecanismos de autenticación mediante doble factor, vienen a proporcionar una solución viable para la seguridad de las redes digitales en el hogar. A pesar de ello la sutileza que implican los protocolos de seguridad informática requiere la constante revisión y perfeccionamiento de los mecanismos que se van planteando, a fin de cumplir cada vez mejor las exigencias de seguridad de cada momento y entorno en el que son aplicados.

Los métodos de autenticación robusta para el acceso remoto han sido extensamente desarrollados; y la autenticación por medio de contraseñas sigue siendo vista como uno de los métodos más simples y convenientes, por sus notables beneficios en cuanto al costo de implementación y su facilidad de uso. Para prevenir los ataques que se basan en la interceptación de información transmitida por la red (*eavesdroping attack*), muchos de los esquemas de autenticación modernos se han basado en las contraseñas de un solo uso (OTPs).

Dada la creciente presencia de los dispositivos inalámbricos en las redes, para los cuales es importante el ahorro de consumo energético, se ha generado una tendencia a producir protocolos más ligeros para los entornos en los que puedan intervenir este tipo de equipos. Actualmente existen mecanismos bastante eficientes y seguros de OTPs, como lo son el HOTP -OTP basado en HMAC (*Hash Message Authentication Code*)-, y otros, que son factibles para su aplicación en estos escenarios. Por ello, estas contraseñas dinámicas, en combinación con las tarjetas inteligentes, constituyen una de las formas

más simples y populares de autenticación de doble factor; y han sido ampliamente adoptadas debido a su bajo costo computacional y su conveniente portabilidad.

A pesar de las ventajas de estos esquemas, hay investigaciones que indican que algunos métodos sofisticados pueden extraer secretos y valores de verificación de las tarjetas inteligentes (Kocher, Jaffe, & Jun, 1999; Messerges, Dabbish, & Sloan, 2002). Por lo tanto, la debilidad de los esquemas de autenticación usando tarjetas inteligentes se debe fundamentalmente al hecho de que la información guardada en dicha tarjeta pueda ser utilizada por un adversario para construir un mensaje válido de autenticación que le permita acceder ilegalmente al sistema, suplantando al usuario legítimo, aun sin el conocimiento de la contraseña. Esta es una problemática que no tiene una solución sencilla. Sin embargo, recientemente Vaidya et al. (Vaidya, Park, Yeo, & Rodrigues, 2011) han propuesto un esquema de autenticación robusta para redes residenciales usando tarjetas inteligentes, basado en el algoritmo HOTP. En el protocolo propuesto, los autores han evitado la sincronización de tiempo y el uso de verificadores de contraseña en el servidor de autenticación. Estos afirman además que, a diferencia de los anteriores, el protocolo es seguro contra ataques con robo de la tarjeta inteligente. Este esquema planteado incurre en costos computacionales algo más elevados que esquemas precedentes, en orden de brindar más seguridad. Sin embargo, un trabajo posterior (Kim & Kim., 2011) ha señalado importantes fallas de seguridad del protocolo de Viadya et al. (2011) que contradicen el planteamiento de sus autores sobre el nivel de seguridad de este protocolo.

I.1 Objetivos de la tesis

Con el presente trabajo de tesis nos proponemos alcanzar los siguientes objetivos

Objetivo General: Modificar el protocolo de Vaidya et al. (2011) para obtener un protocolo más seguro y eficiente.

Para conseguir este objetivo general nos hemos propuesto lograr una serie de **objetivos específicos** que exponemos a continuación:

- Realizar un análisis, lo más exhaustivo y detallado posible, de las bondades, deficiencias y vulnerabilidades del protocolo de autenticación robusta mediante *one-time passwords* usando tarjetas inteligentes para redes digitales en el hogar, propuesto por Vaidya et al. (2011): Este análisis permitirá la comprensión en detalle del funcionamiento del protocolo en cuestión, lo cual es de vital importancia para proponer cualquier cambio con el objetivo de mejorar el mismo en cualquier aspecto.
- Reducir el número de operaciones criptográficas del protocolo de Vaidya et al. (2011) por medio del uso de un mecanismo distinto de las *cadenas de hash* durante la *fase de solicitud de acceso al servicio*: Aunque las *cadenas de hash* han sido uno de los mecanismos preferidos de autenticación robusta mediante OTPs, estas resultan costosas en general en cuanto a cantidad de operaciones para al menos alguna de las partes que interviene el en protocolo. Como parte de una de las modificaciones más importantes que nos proponemos, está la sustitución de este mecanismo por uno más eficiente que no sacrifique la seguridad del protocolo con el objetivo de mejorar el rendimiento computacional del mismo.
- Rectificar algunas de las vulnerabilidades de seguridad más relevantes del protocolo de Vaidya et al. (2011): En nuestro trabajo evidenciaremos que el protocolo propuesto por Vaidya et al. (2011) no es tan seguro como han afirmado sus autores, lo que nos lleva a proponer modificaciones al mismo para solucionar algunas de sus vulnerabilidades y deficiencias.
- Proponer un nuevo protocolo más eficiente y seguro que el de Vaidya et al. (2011), como resultado de las modificaciones realizadas: Las modificaciones que se planteen, no solo se justificarán de forma aislada, sino que además se combinarán con la buena estructura y principios de funcionamiento del protocolo de Vaidya et al., para resultar en un nuevo protocolo usando tarjetas inteligentes, para redes digitales en el hogar, más eficiente y seguro.
- Lograr un nivel de seguridad similar al del protocolo de Kim et al. (2011): Basado en los tipos de ataques factibles contra este tipo de esquemas, evidenciaremos que nuestra propuesta alcanza un nivel de seguridad equiparable a la propuesta de

Kim et al. (2011), aportando incluso ventajas desde el punto de vista de la eficiencia por encima de este último.

I.2 Organización de la tesis

El cuerpo restante del presente trabajo de tesis está organizado de la siguiente manera:

En el *Capítulo 1: "Fundamentos Teóricos"*, se analizan las características fundamentales, desde el punto de vista matemático, de los principales elementos relacionados con la criptografía, que se usarán como herramientas indispensables en los mecanismos y protocolos con los que se van tratar en este trabajo. Además se especifican los dos puntos de vista desde los cuales puede ser tratada la seguridad de los mecanismos relacionados con la criptografía. Seguido de esto se introducen las bases conceptuales para la autenticación de entidades. También se describen los tipos de factores a partir de los cuales este proceso puede ser llevado a cabo, haciendo énfasis en la autenticación basada en contraseñas, a la cual se le dedica especial atención. Dentro de esta última temática se revelan las deficiencias y debilidades de las contraseñas estáticas, para así dar paso al estudio de mecanismos de autenticación robusta basados en contraseñas de un solo uso (*One Time Passwords*). Todos estos temas son tratados en este capítulo con un lenguaje sencillo y sin entrar en formalidades matemáticas con todo rigor, ya que nuestra intención es que se entiendan, lo más claramente posible, las propiedades, ideas y conceptos importantes para la comprensión y análisis de los métodos utilizados en los protocolos de autenticación robusta, para lo cual no es imprescindible el conocimiento a fondo de los elementos criptográficos formales.

En el *Capítulo 2: "Aspectos Metodológicos"*, se brindan conceptos e ideas esclarecedoras para el mejor entendimiento de lo que es un protocolo de autenticación de entidades, y se plantean las bases para el análisis y diseño de los mismos. De igual forma se plantea el ámbito general dentro del cual será enmarcado el tema de la seguridad de los protocolos que se exponen. Seguido de esto, se ilustra el proceso paulatino de

construcción de un protocolo seguro, hasta un cierto nivel, mediante la eliminación sucesiva de algunos ataques básicos. Durante este proceso, se introducen notaciones y terminologías que serán utilizadas en el resto de la tesis, así como observaciones generales importantes para el fortalecimiento de la seguridad de los protocolos de autenticación. Todo esto sirve como una guía útil para la reducción de vulnerabilidades en protocolos de autenticación robusta. Luego de ello se realiza un estudio detallado de los parámetros variables con el tiempo, que pueden ser conjugados con las contraseñas estáticas para aportar dinamismo a las mismas, lo cual constituye la solución más efectiva a las vulnerabilidades de este tipo de contraseñas estudiadas en el Capítulo 1. A partir de esto, se propone un nuevo tipo de parámetro variable con el tiempo, que conjuga dos de los tipos de parámetros estudiados con el objetivo de aprovechar las ventajas de cada uno de ellos para brindar un uso más eficiente de los mismos; esta iniciativa no ha sido encontrada en ninguna de las bibliografías revisadas en el presente trabajo y hemos decidido nombrarle *Valor de Estados Semi-Aletorios*. Este tipo de parámetro que hemos introducido servirá posteriormente para mejorar el rendimiento del protocolo de Vaiya et al. lo cual es uno de los objetivos principales de esta tesis. Por último se estudian en este capítulo los diferentes tipos de contraseñas de un solo uso (OTPs) haciendo hincapié en aquellas que usan *cadenas de hash* y luego se describe brevemente el algoritmo de HOTP, ya que estos mecanismos son utilizadas en el protocolo de Vaidya et al. (2011), que tiene especial relevancia para nuestro trabajo.

En el *Capítulo 3: "El protocolo de Vaidya et al. (2011), resultado de la evolución de la autenticación robusta"*, se expone un recorrido evolutivo por del estado del arte de los protocolos de autenticación robusta que usan *One Time Passwords*, hasta llegar al protocolo de Vaidya et al. (2011). En este punto, antes de presentar dicho protocolo se expone el tipo de escenario, la infraestructura de trabajo, y el funcionamiento general del mismo, con el objetivo de brindar todas facilidades posibles para su entendimiento. Finalmente se presenta una descripción detallada del protocolo de Vaidya et al., en la cual se muestran los mensajes intercambiados y la forma de actuar de cada ente principal a lo largo de cada fase de ejecución del protocolo.

En el Capítulo *4: "Análisis y Discusión de los Resultados"*, se analizan las bondades y deficiencias más importantes del protocolo de Vaidya et al. (2011). Este análisis integral se basa en gran parte en las pautas delineadas en el Capítulo 2, así como en la metodología de reducción de vulnerabilidades planteada en este mismo capítulo. Como resultado de ello se identifican las deficiencias de rendimiento y vulnerabilidades de seguridad más importantes del protocolo de Vaidya et al. Además se analizan algunas modificaciones ya propuestas por Kim y Kim (2011) al protocolo en cuestión, las cuales, aunque resuelven algunos problemas, lo hacen a costa la disminución de la eficiencia del protocolo de Vaidya et al. Todos estos análisis y las ideas planteadas durante el capítulo 2 dan lugar a las modificaciones que hemos realizado al protocolo de Vaidya et al. (2011), con el objetivo de obtener un protocolo más seguro, pero a la vez más eficiente, las cuales se plantean y se argumentan en este capítulo. Una descripción detallada del protocolo propuesto, como consecuencia de estas modificaciones, también es incluida en esta parte. Posteriormente se realiza un análisis de la seguridad de nuestro protocolo, a partir de los ataques factibles contra el mismo. Seguido de esto, se compara nuestro protocolo con el de Vaidya et al., haciendo hincapié en la notable reducción de las operaciones criptográficas que se logra con nuestras modificaciones, lo cual ilustramos a través de graficas, y de igual modo, se incluye una tabla comparativa del nivel de seguridad entre nuestra propuesta, el protocolo de Vaidya et al. y el protocolo de Kim et al., en base a los ataques presentados en los acápites 4.3 a) y 4.3 b).

Luego se exponen las *Conclusiones*, en las cuales se resumen los resultados y logros de nuestro trabajo, teniendo en cuenta los objetivos que nos hemos planteado.

Finalmente se plantean las *Recomendaciones y Trabajos Futuros*, donde se exponen los retos y posibles líneas de investigación que podrían dar continuidad a nuestro trabajo.

CAPÍTULO 1: FUNDAMENTOS TEÓRICOS

1.1 Antecedentes

Desde tiempos antiguos se han desarrollado mecanismos de protección de la información los cuales comenzaron por hacer un uso importante de medidas físicas. En las últimas décadas, con el cambio radical de la forma en se transmite la información, la cual a migrado hacia el formato electrónico, se hace mucho más fácil la capacidad y habilidad para copiar y manipular la misma sin que esto sea advertido. El principal impulso para el desarrollo de técnicas criptográficas avanzadas ha estado durante muchos años movido por las esferas militares y gubernamentales, dado el interés por la protección de los secretos nacionales. Sin embargo, la rápida proliferación y popularización de los ordenadores personales desde los años 1960s ha traído consigo la demanda del sector privado de proteger información digital para brindar servicios seguros. De este modo, la criptografía ha jugado, desde hace mucho tiempo, un papel fundamental como una herramienta que proporciona seguridad a la información de forma independiente a los medios físicos donde esta es almacenada o trasmitida.

En este capítulo nos basaremos en algunos elementos matemáticos generales, para introducir ciertos conceptos, terminología y planteamientos más específicos, relacionados con la criptografía y la seguridad de la información; sin los cuales no pueden ser comprendidos los algoritmos, análisis y planteamientos que se hacen en el presente trabajo, y que a la vez, dejan ver la complejidad y dificultad de los temas tratados.

Definición 1.1: La Criptografía es el estudio de técnicas matemáticas relacionadas a aspectos de la seguridad de la información, tales como confidencialidad, integridad de datos, autenticación de entidades y autenticación de origen de los datos (Menezes, Oorschot, & Vanstone, 1997).

La seguridad de la información se manifiesta de diferentes modos en dependencia del contexto y los requerimientos. Sin importar quién esté involucrado, las partes que

intervienen en una transacción necesitan confiar en que ciertos objetivos relacionados a la seguridad de la información son cumplidos durante el intercambio de la misma.

Expliquemos los 4 objetivos centrales que persigue la criptografía (Menezes et al., 1997).

- *Confidencialidad:* Es un servicio usado para mantener el contenido de una información legible solo para aquellos que están autorizados a acceder a ella, y sin sentido útil para aquellos sin esta autorización. Secrecía es un término utilizado que es sinónimo de lo planteado.
- *Integridad de Datos:* Es un servicio que evita la modificación no autorizada de datos. Para lograr este objetivo es necesario tener la habilidad de detectar manipulación de datos por partes no autorizadas, donde la manipulación incluye: inserción, eliminación y sustitución.
- *Autenticación:* Es un servicio relacionado con la identificación, el cual se aplica tanto a entidades como a información en sí. Dos partes que establecen una comunicación deben identificarse mutuamente. Igualmente, la información que se entrega durante una comunicación debe ser autenticada teniendo en cuenta parámetros como el origen, la fecha de emisión, el contenido, el tiempo de envío, etc. Por estas razones este aspecto de la criptografía se subdivide en dos clases fundamentales: *autenticación de entidades* y *autenticación de datos*. La autenticación de origen de datos incluye la integridad de datos ya que si estos han sido modificados entonces la fuente ha cambiado.
- *No-Repudio:* Es un servicio que previene que una entidad niegue un compromiso asumido o ciertas acciones llevadas a cabo por esta. Esto proporciona un medio para resolver disputas en las que una entidad niega que ciertas acciones tuvieron lugar. Estos casos, por lo general, requieren la intervención de una tercera parte de confianza.

Podríamos decir que la criptografía se ha creado para prevenir y detectar el engaño y otros tipos de actividades maliciosas relacionadas con el manejo de información y la comunicación de la misma. Para lograr estos objetivos se usan una serie de herramientas criptográficas básicas a las cuales se les suele llamar *primitivas*.

Estas primitivas son: las *funciones hash* de longitud arbitraria, las funciones de un solo sentido, las sucesiones aleatorias y *pseudo-aleatorias*, los *cifradores* de clave simétrica, *cifradores* de clave pública, las firmas digitales, las primitivas de identificación.

Estas primitivas se pueden evaluar con respecto a varios criterios los cuales son (Menezes et al., 1997):

- *Nivel de Seguridad:* que es, por lo general, un factor difícil de cuantificar. Para medirla se suele usar el número de operaciones que un atacante tiene que utilizar para vulnerar el sistema. Típicamente se define con una cota superior de la cantidad de trabajo necesaria para derrotar el objetivo, a lo cual se suele llamar *factor de trabajo (work factor).*
- *Funcionalidad:* Por lo general varias primitivas deben ser combinadas para alcanzar los objetivos de seguridad requeridos. En dependencia de las propiedades básicas de las mismas estas resultarán más o menos efectivas y operables dentro del esquema donde son puestas en práctica.
- *Métodos de operación:* Las primitivas pueden ser aplicadas en diferentes formas y con diferentes entradas, produciendo diferentes tipos de funcionalidad en sus modos de operación.
- *Rendimiento:* Se refiere a la eficiencia de una primitiva en un modo particular de operación, y puede ser medido en diferentes formas; siendo entre las más usuales el conteo de cantidad de operaciones y el costo computacional por unidad de tiempo.
- *Facilidad de Implementación:* Se refiere al nivel de dificultad que implica llevar a cabo una instancia práctica de una primitiva dada. Esto puede incluir la complejidad de implementación de la primitiva en software o hardware.

El nivel de importancia de los diferentes criterios de evaluación de las primitivas criptográficas es relativo, y varía de acuerdo al tipo de aplicación de las mismas y los recursos de que se disponen en el entorno donde son puestas en práctica. Por ejemplo, en un entorno donde el consumo de potencia es limitado, puede ser preferible sacrificar un poco el nivel de seguridad por un mejor rendimiento del sistema en general.

Para el presente trabajo solo nos importa el uso de ciertas primitivas las cuales nos restringiremos a estudiar brevemente. En particular nos interesan aquellas primitivas que se usan en el contexto de la criptografía de clave simétrica y que proporcionan un rendimiento aceptable en su aplicación, ya que éstas contribuyen a la reducción del costo computacional de las operaciones criptográficas involucradas, lo cual es primordial para el cumplimiento de los objetivos de nuestro trabajo.

1.2 Funciones de un solo sentido (One-Way functions)

Existen ciertos tipos de funciones que juegan un papel muy importante en la criptografía como son las funciones de un solo sentido.

Sacrificando un poco el rigor en busca de simplicidad, daremos a continuación una definición, casi intuitiva, de las mismas.

Definición 1.2: Una función f, de un conjunto X en un conjunto Y, es llamada $función\ de\ un\ solo\ sentido$ si $f(x)$ es "*fácil"* de computar para todos los $x \in X$, pero para "*esencialmente todos"* los elementos $y \in Im(f)$ no es "*computacionalmente factible"* encontrar algún $x \in X$ tal que $f(x) = y$ (Menezes et al., 1997).

Aclaración sobre los términos usados:

i. Aunque una definición precisa de "*fácil"* y "*computacionalmente factible"* es necesaria para una definición más rigurosa, por ahora nos interesa mantener la simplicidad para un mejor entendimiento del objetivo de este tipo de primitiva. De cualquier forma estos términos siempre deben ser interpretados en relación a un marco de referencia relacionado con el contexto donde son usados.

ii. Para nuestros objetivos prácticos es suficiente aclarar que "*fácil"* suele referirse al consumo *polinomial* de tiempo y espacio, o a algo que suele resolverse con no muchas operaciones de cómputo, o dentro de un tiempo corto.

iii. El término "*no factible computacionalmente"* suele referirse a un esfuerzo *no-polinomial (NP),* o un esfuerzo que exceda los recursos que se entiende que están disponibles para un atacante; puede ser especificado mediante una cota inferior

para el número de operaciones necesarias, o la memoria requerida, en términos de un parámetro de seguridad dado, o asociado al hecho de que la probabilidad de que una propiedad sea violada es exponencialmente pequeña.

iv. La frase "*para esencialmente todos los elementos* $y \in Im(f)$ *no es computacionalmente factible encontrar algún* $x \in X$ *tal que* $f(x) = y$" se refiere a que solo hay unos pocos valores en $Im(f)$ para los cuales sería "*fácil*" encontrar un valor x tal que $y = f(x)$. De otro modo, existen funciones en la cuales, se puede encontrar el inverso fácilmente para unos pocos valores, no ocurriendo lo mismo con el resto de los valores. Otra forma apropiada de describir esta propiedad de las funciones de un solo sentido es la siguiente: para un aleatorio $y \in Im(f)$ no es *computacionalmente factible* encontrar un $x \in X$ tal que $f(x) = y$.

Las funciones de un solo sentido son usadas en las aplicaciones criptográficas donde no es necesario descifrar la información, sino que lo que interesa es establecer una correspondencia entre un mensaje M y un código $C = f(M)$ que es usado como un identificador de M para un sistema dado.

Definición 1.3: Una *función trampa unidireccional* es una función $f: X \to Y$ *de un solo sentido,* con la propiedad de que, dada cierta información adicional (llamada *información trampa*) se vuelve factible para cualquier $y \in Im(f)$ encontrar un $x \in X$ tal que $y = f(x)$ (Menezes et al., 1997).

Además de las funciones trampa, existe un tipo muy especial de funciones las cuales son llamadas *funciones hash* y que constituyen una de las primitivas fundamentales. A las *funciones hash* le prestaremos especial atención en el presente trabajo, dado que de ellas depende en gran parte la seguridad de los protocolos de autenticación con los que vamos a tratar posteriormente.

(a) Funciones hash

Definición 1.4: Una *función hash* es una función computacionalmente eficiente, que mapea cadenas de longitud arbitraria a una cadena de longitud fija, llamada *valor-hash* (Menezes et al., 1997).

La idea básica es que un *valor-hash* funciona como una representación compacta de una cadena de entrada.

Notemos que dado que el conjunto de las posibles entradas (dominio) de una *función hash* H cualquiera es considerablemente mayor que el rango de la misma (conjunto imagen), esta no puede ser inyectiva, por lo que necesariamente existen valores x, y con $x \neq y$ tales que $H(x) = H(y)$. Además de esto se tendrá que $H(x) = H(y) \Rightarrow x = y$ con alta probabilidad. El primer hecho matemático es denominado "colisión" cuando se trata con estas funciones, y es, en conjunto con la segunda afirmación, un aspecto primordial para los diferentes tipos de utilidad práctica que poseen este tipo especial de funciones.

De este modo, las *funciones hash* constituyen una herramienta ampliamente usada en las ciencias computacionales y la ingeniería por ejemplo en: los algoritmos de *checksum,* el cotejado de patrones, tablas de búsquedas, etc..

Sin embargo, estas propiedades no bastan; para que una *funcion hash* tenga utilidad en la criptografía se necesitan ciertas propiedades adicionales.

Por ejemplo, en las aplicaciones anteriores no importa la existencia de colisiones mientras la probabilidad de colisión sea muy baja, sin embargo esta baja probabilidad no garantiza que no puedan encontrarse colisiones fácilmente usando algún tipo de "truco" o método matemático.

Hagamos notar que si se restringe una *función hash* H dada a un domino de entradas de $t - bits$, o sea $H: D \longrightarrow R$ con $|D| = 2^t$ y $|R| = 2^n$, $(t > n)$, si H fuese "*aleatoria*", en el sentido de que todos los *valores-hash* producidos por H fuesen igualmente probables, entonces existirían 2^{t-n} valores de entrada que producirían el mismo valor de salida para cada imagen por H, y dos entradas aleatorias producirían el mismo valor con probabilidad 2^{-n} (independiente de t). Por lo tanto la seguridad que brinda una *función hash* está relacionada directamente con la longitud de los *valores-hash* que esta produce (Menezes et al., 1997).

Para la utilidad práctica de estas funciones en la criptografía, debemos en primer lugar, replantear la propiedad básica relacionada con las colisiones en términos de *factibilidad computacional*. De forma que cada *valor-hash* sea únicamente asociado, *en práctica,* con un valor de entrada, y sea muy *costoso computacionalmente* encontrar colisiones. Así, a partir de ahora, cualquier función hash H que usemos en este trabajo deberá cumplir, como mínimo, las siguientes tres propiedades fundamentales.

1) *Pre-Image Resistence (One-Way)*: para, esencialmente, todos los *valores-hash* "y" del rango de H, no es *computacionalmente factible* encontrar algún valor "x" de entrada que lo genere. O sea que, dado cualquier y para el cual no se conoce su correspondiente valor pre-imagen, no es prácticamente posible encontrar ninguna pre-imagen "x" tal que $H(x) = y$. Esto se puede resumir diciendo que la función es *computacionalmente no invertible o unidireccional.*
2) *2nd Pre-Image Resistence (Weak Collision Resistance)*: Dado un valor de entrada x, con $H(x) = y$, no resulta computacionalmente factible encontrar un segundo valor preimágen $x' \neq x$ que produzca el valor y, esto es, tal que $H(x') = H(x) = y$.
3) *Collision Resistance (Strong Collision Resistance)*: *No es computacionalmente factible* encontrar un par (x, x') tal que $H(x) = H(x')$. En este caso siendo x, x' valores de libre elección.

La propiedad 1) coloca a las *funciones hash* que la cumplen dentro de las *funciones de un solo sentido*, y es primordial para brindar la debida seguridad cuando el mensaje de entrada es usado en combinación con algún secreto, impidiendo que un atacante pueda invertir el *valor-hash* resultante y obtener el secreto en claro o una pista útil sobre este.

La propiedad 2) previene contra la posibilidad de que un atacante pueda construir un mensaje falso con el mismo *valor-hash*; brindando así una herramienta de protección contra la modificación indebida o sustitución de datos.

La propiedad 3) provee resistencia a un tipo de ataque específico contra las *funciones hash* llamado "*the birthday attack*", el cual se aprovecha de la existencia inevitable de colisiones en las *funciones hash*, como consecuencia de que estas no son *inyectivas*.

Dados los planteamientos anteriores, es importante que se entienda que existe una diferencia sustancial entre el hecho de la existencia de colisiones y la posibilidad real de encontrar alguna.

Los usos criptográficos más comunes de las *funciones hash*, se encuentran en: las firmas digitales, la integridad de datos y la autenticación de mensajes. Siendo de nuestro interés solo los dos últimos usos.

Para resistir ataques criptográficos más avanzados, las *funciones hash* requieren satisfacer otras propiedades adicionales que garantizan la fortaleza local de estas funciones, asegurando así que una tal *función hash*, como función de todos los bits de una cadena dada, no debe ser débil con respecto a alguna de las partes de sus valores de entrada o de salida. Estas propiedades, necesarias para preservar la seguridad local de las *funciones hash*, son las siguientes.

4) *Non-Correlation*: Los bits de entrada y los de salida por la *función hash* no deben estar correlacionados. Esto relaciona a las *funciones hash* que lo cumplan con los buenos *cifradores* de bloque, donde cada bit de entrada afecta cada bit de salida.
5) *Near-Collision Resistance*: Debe ser *difícil* encontrar un par de valores de entrada x, x' tales que sus correspondientes valores de salida $H(x)$, $H(x')$ difieran solo en un pequeño número de bits.
6) *Local One-Wayness:* Recobrar cualquier sub-cadena de un valor pre-imagen, ha de suponer el mismo nivel de dificultad que recobrar la cadena completa.

Una forma de lograr la propiedad 5) es garantizando que si el valor de entrada cambia en uno o más bits, el *valor-hash* de salida debe cambiar en aproximadamente la mitad de los bits.

De una *función hash* que cumpla las propiedades desde la 1) a la 6), se puede decir que es estadísticamente fuerte, global y localmente. Estas propiedades le dan a una *función hash* la resistencia que se necesita para las aplicaciones criptográficas ante la manipulación de los valores de las mismas, a lo cual se le suele llamar *no-maleabilidad*. A partir de ahora, y dentro del contexto del presente trabajo, llamaremos a una tal función: "*función hash segura"*.

Estas funciones en sí, no requieren el uso de una clave secreta, sin embargo, se pueden combinar con algún secreto para brindar más seguridad en algunos de sus usos prácticos. Cuando estas son usadas para detectar si un mensaje dado ha sido alterado, se obtiene lo que se llama un código MDC (*Modification Detection Code*). De otro modo, cuando son usadas en combinación con una clave secreta, con el objetivo de proveer integridad de los datos y a la vez autenticación del origen, se obtiene lo que se llama un código MAC (*Message Authentication Code*).

Un código MAC (*Message Authentication Code*) es una familia de funciones parametrizadas por una clave K tales que $MAC_K(M)$ transforma un mensaje arbitrario en otro de longitud fija, de forma tal que se satisfacen las siguientes propiedades (Colin & Anish, 2003):

1. Es computacionalmente fácil calcular $MAC_K(M)$ dados K y M
2. Dado cualquier número de valores MAC para un valor de K determinado, es computacionalmente difícil encontrar un valor de MAC válido para cualquier mensaje nuevo, sin el conocimiento de K.

1.3 HMAC (Hash-MAC)

Existen diversas variantes para componer un código MAC; en nuestro caso nos interesa estudiar aquella en que se usa una *función hash* junto a la clave secreta; ya que este tipo de implementaciones tienen la ventaja de usar algoritmos criptográficos gratuitos, y simples de usar, incurriendo en costos computacionales mínimos. Por otra parte HMAC es la base para un mecanismo de generación de contraseñas OTP, que será una parte fundamental del protocolo principal que analizaremos en este trabajo.

Uno de los métodos de autenticación de mensajes basado en *funciones hash*, más exitosos, propuestos hasta la actualidad, es el HMAC (*Hash-based Message Authentication Code*), el cual puede ser utilizado para proveer *integridad de datos* y *autenticidad de mensajes*. El mismo está descrito en Krawczyk, Bellare, & Canett (2004).

Esta variante de MAC puede ser utilizada en combinación con cualquier *función hash* segura como lo son MD5 y SHA-1, o cualquier otra más actual. Sin embargo, debe tenerse en cuenta que la seguridad de HMAC está estrechamente ligada a las propiedades criptográficas de la *función hash* subyacente y la fortaleza de la misma.

Los principales objetivos de este tipo de implementación de MAC (Krawczyk et al., 2004) que la distinguen del resto son:

- Usar, sin modificaciones, las *funciones hash* disponibles. En particular aquellas que sean eficientes en software, cuyo código es libre y ampliamente difundido.
- Preservar la eficiencia original de dicho tipo de funciones, sin que esta se vea significativamente degradada en su uso.
- Usar y manejar de un modo simple las claves que intervienen.
- Poder propiciar un análisis criptográfico claro, de la fortaleza del mecanismo de autenticación a partir de propiedades razonables asumidas para la *función hash* subyacente.
- Dar la posibilidad de reemplazar fácilmente la *función hash* subyacente en caso de que alguna más rápida o más segura disponible sea requerida.

La definición del algoritmo de HMAC es como sigue: sea H una *función hash* iterativa que trabaja internamente con bloques de longitud b, HMAC actúa sobre entradas x de longitud arbitraria y usa una clave aleatoria K, de longitud $l \leq b$, calculándose del siguiente modo:

$$HMAC_K(x) = H\big((\overline{K} \oplus opad) \parallel H((\overline{K} \oplus ipad) \parallel x)\big)$$

Donde $\overline{K}$ es el completamiento de K añadiendo $0's$ para formar un bloque de longitud b-bits; $opad$ e $ipad$ representan cadenas fijas de b-bits, dados en función de la clave, y están formados por la repetición del byte x $'36'$ y x $'5c'$ de K respectivamente (Evans, Bond, & Bement, 2002).

El diseño de este mecanismo estuvo motivado por la existencia de ataques a otras formas más triviales de combinar una *función hash* con una clave secreta. La operación externa de la *función hash* enmascara el resultado intermedio de la aplicación interna de la misma en conjunto con la clave. Los valores *opad* (*outer-pad*) e *ipad* (*inner-pad*) que

actúan de forma externa e interna respectivamente, están concebidos para proporcionar una distancia de *Hamming* suficientemente larga, de modo que las claves interna $K_1 = \bar{K} \oplus ipad$ y externa $K_2 = \bar{K} \oplus opad$ tengan pocos bits en común. Como resultado de todas estas medidas, se obtiene un fortalecimiento del mecanismo de MAC, de forma tal que si se usa una "buena" *función hash* criptográfica, los ataques existentes contra las *funciones hash* en combinación con claves secretas, serán muy poco prácticos.

1.4 Generación de números aleatorios y pseudo-aleatorios

La seguridad de numerosos sistemas criptográficos depende de la generación de cantidades impredecibles, por ejemplo: la generación de claves para distintos mecanismos de cifrado, la generación de primos para el algoritmo RSA, los mecanismos de desafío-respuesta utilizados en diversos protocolos de autenticación. La impredictibilidad de los secretos está estrechamente ligada al nivel de incertidumbre y aleatoriedad de los mismos. De este modo, los parámetros secretos de los algoritmos criptográficos deben ser suficientemente grandes y aleatorios en el sentido probabilístico, para descartar la posibilidad de que un adversario pueda recurrir a una estrategia de búsqueda factible para adivinar valores generados. En este sentido, *el nivel de aleatoriedad* con que se generan los secretos es un factor fundamental para la seguridad de los mismos (Menezes et al., 1997). Sin embargo, en ciertos casos prácticos, la generación de valores aleatorios puede ser ineficiente y no muy efectiva, para cumplir el objetivo señalado; lo cual depende de la forma en que dichos valores son generados.

Definición 1.5: Un *generador de bits aleatorios* es un dispositivo o un algoritmo que es capaz de producir secuencias de bits *estadísticamente independientes* (la probabilidad de que la fuente emita un valor 0 o 1 no depende de los bits previamente emitidos) y *balanceados* (la probabilidad de que la fuente emita un valor 1 es $1/2$).

Estos generadores de bits aleatorios son a su vez usados para la generación de números aleatorios (uniformemente distribuidos) en un cierto intervalo $[0, n]$.

Un verdadero generador aleatorio de bits requiere una fuente natural de aleatoriedad. En diversas aplicaciones de las sucesiones aleatorias es suficiente con esto, pero en el caso de la mayoría de las aplicaciones criptográficas es importante que el generador no pueda estar sujeto a la observación o manipulación de un adversario. Los generadores de bits basados en fuentes naturales de azar pueden verse influenciados por factores externos y por tanto no funcionar adecuadamente.

Para la generación de bits aleatorios existen dos tipos fundamentales de fuentes:

- *Generadores basados en Hardware*: explotan la aleatoriedad relacionada con ciertos fenómenos físicos estos en ciertos casos pueden producir bits de forma correlacionada o con *bias*, por lo que se suelen someter luego a procedimientos de corrección. Algunos ejemplos de tales fenómenos físicos son:
  - *i. el tiempo transcurrido entre la emisión de partículas durante una desintegración radiactiva;*
  - *ii. el ruido térmico de un diodo semiconductor o resistencia;*
  - *iii. la inestabilidad de la frecuencia de un oscilador de funcionamiento libre;*
  - *iv. la turbulencia de aire dentro de una unidad de disco sellado que causa fluctuaciones aleatorias en los tiempos de latencia de lectura de un sector del disco;*
  - *v. el sonido de un micrófono o una entrada de vídeo desde una cámara;*
- *Generadores basados en Software*: el diseño de este tipo de generadores suele ser mas difícil que en hardware. Algunos de los procesos en los cuales se pueden basar son:
  - *i. el reloj del sistema;*
  - *ii. los tiempos transcurridos entre pulsaciones de teclas o movimientos del mouse;*
  - *iii. contenido de los buffers de entrada-salida;*
  - *iv. entradas del usuario;*
  - *v. valores del sistema operativo, como pueden ser la sobrecarga del sistema y las estadísticas de red.*

El comportamiento de estos procesos puede variar considerablemente de acuerdo con varios factores en dependencia de la plataforma computacional sobre la que son ejecutados. También suele ser complicado evitar la posibilidad de que un adversario observe o manipule estos procesos. De modo que es recomendable utilizar tantas fuentes de aleatoriedad como sea posible, para un diseño satisfactorio. Con esto se brinda fortaleza al mecanismo ante la posibilidad de que algunas de estas fuentes fallen o bien sean observadas o manipuladas por un adversario.

Para generar cadenas de bits aleatorias de forma más eficiente se suelen utilizar los llamados *generadores de bits pseudo-aleatorios*.

Definición 1.6: Un *generador de bits pseudo-aleatorios* es un algoritmo determinista, el cual, dada una secuencia de bits realmente aleatorios de longitud k, produce una secuencia binaria de longitud $l \gg k$ de apariencia aleatoria. La secuencia de entrada suele llamarse *semilla (seed)*.

Aunque en este caso la secuencia de salida por tales generadores no es realmente aleatoria, la idea de los mismos es extender una secuencia aleatoria a otra mucho mayor, de forma tal que un adversario no pueda eficientemente distinguirla de una realmente aleatoria de la misma longitud. Un requerimiento importante a tener en cuenta para la seguridad en la generación de estas secuencias binarias *pseudo-aleatorias*, es que la longitud de la semilla aleatoria debe ser suficientemente larga como para que no sea factible al adversario encontrar la misma mediante una búsqueda en el conjunto de todas las semillas posibles. Igualmente, es importante que los bits de salida sean impredecibles para un adversario con limitados recursos computacionales.

Con el objetivo de analizar la confiabilidad de distintos tipos de generadores de bits pseudo-aleatorios existen variedad de pruebas estadísticas a las cuales estos deben ser sometidos, para detectar si los valores generados cumplen con las características esperadas de una secuencia realmente aleatoria.

Así por ejemplo, se puede decir que un *generador de bits aleatorios pasa todas las pruebas estadísticas de tiempo-polinomial*, si ningún algoritmo de tiempo-polinomial puede distinguir entre una secuencia de salida del generador y una secuencia realmente aleatoria con probabilidad significativamente mayor que $1/2$.

1.5 Criptografía de clave simétrica vs criptografía de clave pública.

Una vez vistas algunas primitivas criptográficas elementales, y señalados algunos aspectos relevantes sobre la generación de valores aleatorios, nos interesa ver algunas características generales de los dos tipos fundamentales de esquemas de cifrado: los de clave simétrica y los de clave privada. Una comparación entre estos dos grandes grupos de cifrados nos permitirá conocer las utilidades y conveniencias de cada uno sobre el otro.

Un esquema de cifrado, en una comunicación bipartita, usando clave simétrica puede ser descrito por el diagrama de bloque de la Fig. 1 que se muestra a continuación (Menezes et al., 1997).

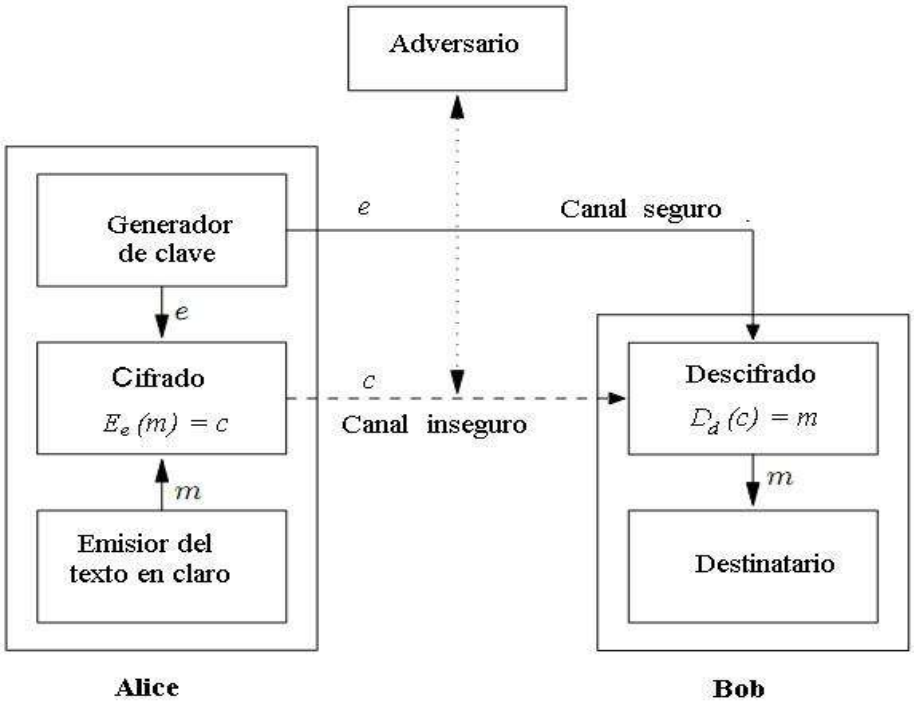


Figura 1: Esquema de cifrado, en una comunicación bipartita, usando clave simétrica.

Uno de los problemas fundamentales de este tipo de esquemas criptográficos es el cómo lograr compartir la clave secreta de forma segura, lo cual es usualmente llamado, *problema de distribución de claves (key distribution problem)*.

En un criptosistema cualquiera, ya sea de clave simétrica o de clave pública, se debe cumplir que es necesario tener la clave de descifrado para poder recobrar el texto en claro a partir del texto cifrado. De modo que, aún si el atacante tiene completo conocimiento de los algoritmos usados (los cuales por lo general son públicos), y otras informaciones adicionales, no le debe resultar factible recuperar el texto en claro sin la clave. De hecho, siempre supondremos que para un adversario del sistema el único secreto es la clave del mismo; esto es conocido como el *principio de Kerckhoff* (Kerckhoffs, 1883).

El tamaño del espacio de llaves es el número de pares de llaves de cifrado/descifrado disponibles en el sistema de cifrado y, en gran medida, la seguridad del esquema de cifrado está relacionado con la extensión de este espacio.

Una condición necesaria, aunque no suficiente, para que un esquema de cifrado sea seguro, es que el espacio de claves sea suficientemente grande como para que no sea factible una búsqueda exhaustiva.

Los esquemas de criptografía simétrica y criptografía pública tienen varias ventajas y desventajas, algunas de las cuales son comunes a ambos tipos de esquemas.

(a) Ventajas de la criptografía de clave simétrica (Menezes et al., 1997):

1. Los *cifradores* de clave simétrica pueden ser diseñados para proporcionar una alta velocidad de procesamiento de datos. Algunas implementaciones en hardware alcanzan hasta cientos de megabytes por segundo, mientras que las implementaciones en software pueden alcanzar hasta una velocidad de megabytes por segundo.
2. Las claves necesarias para una seguridad aceptable en *cifradores* simétricos son relativamente cortas.
3. Los *cifradores* de clave simétrica pueden ser utilizados como primitivas para construir varios mecanismos criptográficos como por ejemplo: números aleatorios, funciones hash, y esquemas eficientes de firma digital, entre otros.

4. Los *cifradores* de clave simétrica pueden componerse para construir *cifradores* más robustos.

(b) Desventajas de la criptografía de clave simétrica (Menezes et al., 1997):

1. En una comunicación bipartita, la clave debe permanecer secreta en ambos extremos.
2. En una red amplia se necesitarían demasiados pares de claves para manejar las comunicaciones de forma segura entre cada par de participantes de la red. Esto hace necesario la intervención de una tercera parte de confianza incondicional (*unconditionally trusted third party-TTP*).
3. En una comunicación bipartita usando clave simétrica entre dos entes se debe cambiar la clave con cierta frecuencia, siendo ideal que se use solo una clave para cada sesión de comunicación.
4. Los mecanismos de firma digital que se basan en clave simétrica generalmente requieren claves bastante extensas o el uso de una tercera parte de confianza (*TTP*).

(c) Ventajas de la criptografía de clave pública (Menezes et al., 1997):

1. Solo la clave privada debe ser mantenida en secreto (aunque la autenticidad de la clave pública debe garantizarse).
2. La administración de claves en la red requiere la intervención de una tercera parte confiable funcional, no necesariamente incondicional, como es el caso de la criptografía de clave simétrica.
3. Dependiendo del modo de uso, el par de claves privada/pública pueden mantenerse iguales por un largo período de tiempo, o por varias sesiones.

4. Muchos esquemas de clave pública llevan a mecanismos eficientes de firmas digitales con claves de menor longitud que en los correspondientes mecanismos de clave simétrica.
5. En redes extensas, el número de claves necesarias en los esquemas de clave pública, resulta considerablemente menor que en el caso de la criptografía de clave simétrica.

(d) Desventajas de la criptografía de clave pública (Menezes et al., 1997):

1. La velocidad de encriptación en el caso de del los esquemas de cifrado con clave pública son de varios órdenes de magnitud menor que los de clave simétrica.
2. La longitud de las claves es normalmente bastante mayor que en los esquemas de clave simétrica.
3. No se ha establecido mediante ninguna prueba formal que alguno de los esquemas de criptografía de clave-pública sea segura y los esquemas de este tipo más eficientes basan su seguridad en la presunta dificultad de resolución de ciertos problemas de la teoría de números y la matemática discreta.

De todo lo enunciado anteriormente, la ventaja fundamental que más nos interesa de la criptografía de clave simétrica sobre la de clave pública es su eficiencia. Por otra parte, la criptografía de clave pública tiene la ventaja de que no se necesita un intercambio de clave secreta, sin embargo, requiere una infraestructura de clave pública.

Lo más sensato para la seguridad es tomar lo mejor de cada tipo de esquema y usar un esquema mixto. Esto es realmente posible, y la forma de lograrlo es usando la clave de publica del esquema asimétrico para el intercambio de la clave simétrica secreta y esta última es entonces usada para cifrar toda la información.

Los métodos de cifrado/descifrado (de la criptografía de clave simétrica) se dividen en dos grandes grupos:

- *Cifrado en flujo*: estos algoritmos realizan el cifrado bit a bit de la información de entrada. En este caso se parte de la clave para componer una cadena más larga la

cual se usa para cifrar con la operación XOR. Se pueden considerar una generalización del cifrado *One Time Pad* (que luego analizaremos en detalle).

- *Cifrado en bloque*: Funcionan como una tabla de códigos, donde la clave secreta determina la codificación. Emplean tanto la confusión como la difusión. Los *cifradores* en bloque dividen el texto en bloques de longitud fija y obtienen el texto cifrado iterando una función F un cierto número de veces.

De modo general los *cifradores* de bloque son más fáciles de optimizar en software, mientras que los *cifradores* de flujo suelen ser más eficientes en hardware.

Dado que uno de nuestros principales intereses está dirigido hacia la eficiencia computacional, obviaremos los esquemas de autenticación que usan criptografía de calve pública, de modo que aquellos esquemas de autenticación con los que trataremos en esta tesis usarán operaciones criptográficas ligeras basadas en las primitivas que se describen en este capítulo y, a lo sumo, usarán en algún momento criptografía de clave simétrica en caso de que sea necesario.

1.6 Confidencialidad perfecta y seguridad práctica

Existe una diferencia sustancial entre la seguridad teórica y la seguridad práctica, lo cual nos lleva a distintos tipos de modelos para evaluar la seguridad de los mecanismos criptográficos. La forma práctica de medir la seguridad está basada en las métricas computacionales relacionadas generalmente con la complejidad y el costo computacional. Por esto, para decir que un esquema criptográfico es seguro desde el punto de vista práctico se suele decir que es computacionalmente seguro. Nótese que este es el modo en el cual hemos estado tratando la seguridad de los mecanismos planteados anteriormente. De este modo, la seguridad práctica está relacionada con el hecho que no le sea factible a un atacante derrotar el sistema, con los recursos de que dispone. En estos casos podemos asumir que un atacante no invertirá más valor en recursos para atacar un sistema informático, del valor que tienen para él sus objetivos. La seguridad teórica está, más bien, relacionada con lograr que no se filtre, en los mecanismos de cifrado, alguna

información que pueda ser aprovechada por un adversario; esta última puede ser analizada en ciertos casos usando herramientas probabilísticas.

Para comprender mejor estos dos modelos es importante entender la idea central en la que se basa cada uno, esto es: la Seguridad Computacional (en al caso del Modelo Práctico) y la Seguridad Incondicional (en el caso del Modelo Teórico).

Seguridad Computacional: En este caso se asume que el sistema ha sido bien estudiado para determinar cuáles son los ataques más relevantes, y se mide la cantidad de esfuerzo computacional requerido para llevar a cabo los mejores métodos que existen para derrotarlo. En este sentido un esquema se dice que es *computacionalmente seguro* si se estima que el nivel computacional requerido para derrotarlo (usando el ataque mencionado) excede, por un margen adecuado, los recursos del atacante.

Seguridad Incondicional: La forma más estricta de ver la seguridad es aquella en la que se supone que un adversario posee recursos computacionales ilimitados. En estas condiciones se asume que el mecanismo de seguridad es seguro si el adversario no puede disponer de suficiente información como para derrotar al sistema, entiéndase por ello que no se filtra ninguna forma de verificar si sus intentos son acertados en mayor o menor medida. Este tipo de sistemas suelen llamarse sistemas de secreto perfecto *(perfect secrecy)*; y están caracterizados, por el hecho de que la incertidumbre sobre el texto en claro, luego de haber observado el texto cifrado, es la misma que sin tener el texto cifrado. O sea, que conocer el texto cifrado no ayudará para nada al atacante que pretenda descubrir el texto en claro.

Los esquemas criptográficos de clave pública, no pueden ser de secreto perfecto, ya que dado un texto cifrado C, el texto en claro puede ser obtenido cifrando todos los textos en claro con la clave pública hasta que el mensaje C es obtenido.

Una condición necesaria para que un esquema de cifrado con clave simétrica sea de secreto perfecto es que la clave de cifrado sea al menos tan larga como el mensaje a cifrar. De lo contrario se podría encontrar un mensaje M del espacio de mensajes, tal que para un mensaje C del espacio de mensajes cifrados, ninguna clave K pudo haber producido C a partir de M, lo que permitiría a un cripto-analista, descartar ciertos mensajes en claro, incrementando sus posibilidades de romper el cifrado.

1.7 Cifrado con One Time Pad

Existe un tipo de cifrado simple y de clave simétrica, que alcanza la seguridad incondicional (*perfect secrecy*) usando una clave de la misma longitud del mensaje una sola vez, este tipo de cifrado es llamado *One Time Pad*. Existen varias formas de realizar el cifrado *One Time Pad* sobre un mensaje, pero en este trabajo solo nos interesa aquel que consiste en la aplicación de una operación XOR entre el mensaje y la clave secreta.

Recordemos algunas propiedades básicas importantes de la operación XOR entre cadenas de bits. Supongamos que A, B, C y K son cadenas de bits genéricas de la misma longitud n, y que O es una cadena de n ceros. Si representamos la operación XOR por el operador $\oplus$. Entonces se tiene que:

i. $(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$ *(propiedad asociativa)*

ii. $A \oplus B = B \oplus A$ *(conmutatividad)*

iii. $A \oplus O = O \oplus A = O$ *(O es el nulo de la operación)*

iv. $A \oplus A = O$ *(cada elemento es su propio inverso)*

La propiedad iv. convierte al operador XOR en un operador con propiedades interesantes para la criptografía ya que esto garantiza que un mensaje puede ser cifrado mediante esta operación, utilizado una clave de la misma longitud del mensaje, y luego se puede utilizar esta misma clave para descifrarlo aplicando nuevamente la operación XOR, como se muestra a continuación:

$$A \oplus K = C \quad \Rightarrow \quad C \oplus K = (A \oplus K) \oplus K = A \oplus (K \oplus K) = A \oplus O = A$$

El cifrado con XOR de una cadena se realiza aplicando la operación XOR bit a bit, por lo que XOR es un cifrador en flujo. El cifrado de mensajes mediante un XOR con una clave secreta fue propuesto a inicios del siglo XX por Gilbert S. Vernam, mientras trabajaba para AT&T (Kahn, 1967). Este tipo de cifrado es de *secreto perfecto* si la elección aleatoria de las claves es de distribución uniforme, de modo que estas resulten realmente aleatorizadas; por lo que bajo estas condiciones, y un uso correcto, es matemáticamente imposible "romper" dicho cifrado.

Usemos un poco de matemática formal para mostrar las propiedades estadísticas que hacen que este tipo de cifrado sea tan fuerte.

Supongamos que la clave de cifrado es escogida con un algoritmo realmente aleatorio de forma que los bits de la misma están totalmente balaceados, de modo que, la probabilidad de que cualquier bit de la clave sea 0 o 1 es exactamente $^1/_2$; y que los bits de un mansaje M a cifrar, de la misma longitud de la clave, no están balanceados, siendo p , $0 < p < 1$ la probabilidad de los bits sean 0 y por tanto $1 - p$ la probabilidad de que sean 1.

Sea $M = m_1 m_2 m_3 \dots m_n$, $K = k_1 k_2 k_3 \dots k_n$ y $C = c_1 c_2 c_3 \dots c_n$; con $C = M \oplus K$ se tiene $c_i = m_i \oplus k_i \ \forall\, i = \overline{1,n}$. Siendo M y K independientes en el sentido probabilístico se puede calcular la probabilidad:

$$P(c_i = 0) = P(m_i = 0 \ \wedge \ k_i = 0) + P(m_i = 1 \ \wedge \ k_i = 1)$$
$$= P(m_i = 0) * P(k_i = 0) + P(m_i = 1) * P(k_i = 1)$$
$$= p * \tfrac{1}{2} + (1 - p) * \tfrac{1}{2} = \tfrac{1}{2}$$

Obteniéndose $P(c_i = 0) = \frac{1}{2} = P(c_i = 1) \ \forall\, i = \overline{1,n}$, lo que implica que el texto cifrado tiene la apariencia de una secuencia aleatoria de bits.

De esto se puede inferir que cada mensaje cifrado es igualmente probable, con lo cual se tiene que el cifrado *One Time Pad* con la operación XOR cumple que la probabilidad de conocer obtener el texto en claro dado que se conoce el texto cifrado es la misma que sin tener la información del texto cifrado, y por tanto, esta no compromete en ningún sentido la seguridad del cifrado; esto lo convierte en un cifrado de *secreto perfecto*.

Sin embargo, el cumplimiento de lo anterior está sujeto a la condición de que las claves en el mecanismo de cifrado son escogidas de forma totalmente aleatoria, o lo que es lo mismo, con probabilidad uniforme, como se ha señalado inicialmente. De lo contrario el conocimiento del texto cifrado podía tener cierta influencia positiva en la probabilidad de conocer el mensaje en claro y por tanto darle un punto de partida a un atacante para realizar un criptoanálisis. Además, este no es el único requerimiento para que este mecanismo sea totalmente seguro en su uso práctico, es importante que no se

reutilice la misma clave dos veces, de lo contrario un atacante podría proceder del siguiente modo:

Supongamos que se envían los mensajes A y B cifrados mediante *One Time Pad* con la misma clave secreta K. Sean $C_a = A \oplus K$ y $C_b = B \oplus K$, un atacante podría interceptar estos mensajes para obtener $A \oplus B = C_a \oplus C_b$, y luego, en dependencia de la entropía de los mensajes, utilizar este resultado para realizar deducciones sobre ambos mensajes A y B.

Algo como esto podría poner en peligro la clave, ya que por ejemplo, un atacante que tenga en su poder el mensaje en claro A y el mensaje cifrado C_a puede proceder a calcular $C_a \oplus A = A \oplus K \oplus A = K$.

De este modo podemos decir que *One Time Pad* es un mecanismo de cifrado totalmente seguro si es usado correctamente, o sea si:

1. Las claves de cifrado utilizadas en el mecanismo son escogidas con probabilidad uniforme (de forma realmente aleatorizada).
2. La clave de cifrado es compartida por una vía totalmente segura y permanece secreta.
3. Ninguna Clave es utilizada dos veces o más.

La propiedad 3 es peligrosa cuando los mensajes cifrados no tienen aspecto aleatorio. En los protocolos con los que trabajaremos posteriormente este tipo de cifrado es usado con claves y mensajes con un alto nivel de aleatoriedad, por lo que esta propiedad no debe preocuparnos.

1.8 Autenticación de entidades

En este acápite veremos diferentes técnicas que permiten a un ente (*Verificador*) asegurarse de que la identidad de otro (*Solicitante*), es genuina. Este tipo de verificación es lo que conocemos como autenticación, y existen diversas técnicas para llevarla a cabo, siendo lo más común que el *Verificador* autentique al *Solicitante* comprobando la

veracidad de algún tipo de mensaje, que demuestre que el solicitante posee algún secreto asociado (por acuerdo) entre el ente *Verificador* y el ente genuino.

La autenticación de entidades debe ser tratada de una forma distinta a la autenticación de mensajes, a pesar de que la última puede usarse en la primera. Una diferencia fundamental entre la autenticación de entidades y la autenticación de mensajes es que, esta última, no necesita proveer garantías respecto a cuando el mensaje fue creado, mientras que en la autenticación de entidades la verificación de la identidad de la parte solicitante suele realizarse en tiempo-real, durante la ejecución de comunicaciones actuales.

Definición 1.7: La autenticación de entidades es el proceso mediante el cual una de las partes se asegura (mediante la adquisición de evidencia corroborativa) de la identidad de un segundo ente involucrado en un protocolo, y de que este efectivamente ha participado en el proceso (Menezes et al., 1997).

La autenticación de entidades puede darse en variedad de escenarios y formas, según las cuales, son más convenientes unos tipos de mecanismos que otros, y no siempre el mismo nivel de seguridad es requerido.

Uno de los objetivos principales de este tipo de autenticación, es facilitar el control de acceso a algún tipo de recurso cuando una cierta identidad está ligada a determinados privilegios. Ejemplos de este tipo de situaciones son: el acceso remoto a cuentas de usuario en un ordenador, extracción de dinero en cajeros automáticos, permisos de accesos mediante puertos de comunicaciones, el acceso a ciertas aplicaciones (de software), acceso físico a ciertas áreas restringidas, etc.

En algunos contextos se suele ver de forma unificada la autenticación y la autorización, sin embargo estas son dos funciones bien separables y distintas, ya que la autenticación se trata de una decisión binaria (se verifica positivamente la una identidad o no), mientras que la autorización trata con decisiones más variadas sobre el tipo de acceso que debe tener el ente autenticado a ciertos recursos o servicios del sistema.

En muchos de los casos se puede asumir, sin pérdida de generalidad, que la autenticación se lleva a cabo entre un ente que llamaremos *Usuario* (o *Cliente*) y un *Sistema* al que en ocasiones llamaremos *Servidor*.

La autenticación también es un requerimiento inherentemente asociado al intercambio seguro de claves. Aunque este es un tema de gran importancia y complejidad, en este trabajo solo nos ocuparemos de la autenticación de entidades y no de la parte relacionada al intercambio seguro claves. Este problema lo daremos por resuelto en nuestro trabajo, asumiendo de vez en cuando que ya existen claves simétricas compartidas de forma segura entre algunas de las partes que intervienen en el protocolo.

(a) Bases para la identificación de entidades

La autenticación de entidades puede ser dividida en tres grandes categorías, dependiendo de los tipos de elementos en que se basa la seguridad. Esto es basado en:

1. *Algo conocido:* Un secreto establecido entre dos o más partes. Ejemplo de esto son las contraseñas de usuario (algunas veces usadas para obtener una clave simétrica), los Números de Identificación Personal (*Personal Identification Number*) y las claves secretas o privadas cuyo conocimiento es demostrado en los protocolos de desafío respuesta.
2. *Algo poseído:* Esto es por lo general un accesorio físico con ciertas credenciales propias, que cumple una función semejante a la de un pasaporte. Entre estos se pueden mencionar las tarjetas de cinta magnética, *chipcards* (tarjetas plásticas del tamaño de una tarjeta de crédito con un microprocesador embebido o un circuito integrado), y dispositivos portables con capacidad de cálculo que son capaces de generar contraseñas que varían con el tiempo (generalmente denominados *Tokens*).
3. *Algo inherente (a cada individuo humano):* En esta categoría están los métodos que hacen uso de una característica física y acciones involuntarias (biométricos), como son las firmas escritas, las huellas digitales, la voz, los patrones de la retina ocular, las formas geométricas de patrones de las manos y patrones específicos de la escritura con el teclado. Esta última categoría se aparta de la criptografía.

(b) Autenticación basada en contraseñas

Hoy día es prácticamente improbable encontrar algún usuario de un ordenador con conexión a Internet que no acumule un número significativo de contraseñas para acceder a distintos tipos de servicios, que son la consecuencia de la rápida expansión del Internet y las tecnologías informáticas. Y no solo en ese ámbito usamos contraseñas, pues se puede decir que esta misma es la función que realiza el número PIN (*Personal Identification Number*) de las tarjetas bancarias, por citar otro ejemplo.

Lo comentado es solo la consecuencia de que el primero de los factores de autenticación es el preferido en modo general, o sea "*algo conocido*", como es el caso de las contraseñas. La explicación de la popularidad de este tipo de método está precisamente en el bajo costo y la conveniencia (o funcionalidad) del mismo con respecto al resto. Para entenderlo mejor, recordemos que generar una contraseña no supone ninguna inversión de recursos materiales, siendo algo libre de costo; sin embargo, no ocurre así con una tarjeta inteligente o un dispositivo biométrico. Esto mismo hace, que sea mucho más rápido, para aquellos que administran los sistemas que requieren autenticación, el proceso de asignar o cambiar una simple contraseña, que configurar una nueva tarjeta inteligente o unos parámetros basados en biometría para un usuario dado.

El esquema de autenticación más simple a considerar de esta clase de método, es donde un ente se identifica directamente con un sistema autónomo para acceder al mismo. El ejemplo más común de ello es el caso de un usuario que se autentica con una computadora usando su contraseña.

En un tal esquema de autenticación, es necesario tener en cuenta una serie de aspectos fundamentales relacionados con la forma de actuar y de cada una de las partes. La parte que solicita ser autenticado, el *Usuario*, debe enviar una información al *Sistema* para que se verifique su identidad. En el caso del esquema en cuestión, cada *Usuario* posee un identificador ID_U y una contraseña P y este enviará al *Sistema* el par $(ID_U\ ,\ P)$. El *Sistema* por su parte, deberá guardar algún tipo de información, sobre cada uno de los usuarios del mismo, que le permita verificar la validez de la identidad presentada por cualquier ente, que pretende ser autenticado con el *Sistema* como usuario legítimo. No es

deseable que el *Sistema* almacene la contraseña secreta de cada *Usuario* en claro, ya que una posible fuga de información podría entonces comprometer a muchos usuarios y al *Sistema* en sí. Como el *Sistema* no necesita conocer la contraseña secreta de cada *Usuario*, sino una información que permita verificar este secreto asociado a cada *Usuario*, lo más sensato es guardar la contraseña en forma cifrada. Por otra parte, no es conveniente cifrar las contraseñas con algún mecanismo reversible, a partir de la cual se puedan obtener las contraseñas de cada usuario; por ello, lo más adecuado es cifrar las contraseñas con una función unidireccional. Así las contraseñas estarán a salvo incluso si el archivo con la información sobre estas es comprometido. De este modo, el sistema debe almacenar un archivo con una lista de pares $\left(ID_U\,, f(P)\right)$, Donde $f(P)$ es una función unidireccional (*One Way*) de la contraseña, que generalmente es una *función hash segura*. Una vez que el supuesto *Usuario* envía al sistema un par $(ID'_U\,,\ P')$ el sistema verifica inicialmente que el identificador de usuario ID'_U es correcto, chequeando que el valor dado está registrado en él *Sistema*, o sea, que existe un $ID_U = ID'_U$, y si es así, entonces procede a calcular $f(P')$ y lo compara con el correspondiente valor $f(P)$ almacenado para este *Usuario*. Si estos valores coinciden entonces el *Usuario* es autenticado positivamente, y es autorizado a acceder al *Sistema* y a los servicios a los que tiene permiso dentro de este. Nótese que en este caso hemos indicado desde el comienzo que el *Usuario* se identifica directamente con el *Sistema*, o sea que la comunicación es presencial, por lo que este no requiere ninguna prueba para confirmar la identidad del *Sistema*; ya que lo podrá reconocer físicamente, como es en el caso de un *Usuario* que se autentica con un ordenador en una red conocida, o cuando nos autenticamos con nuestro ordenador personal.

Es importante aclarar que la función f usada para mantener la contraseña en secreto, deberá ser una función *inyectiva* con ciertas propiedades aleatorias o bien ser una *función hash* con las propiedades requeridas al principio -de la 1) a la 6) mencionadas en 1.2.a)-, para que se pueda garantizar con una muy alta probabilidad que $f(P') = f(P) \Rightarrow P' = P$, y que no sea factible para un atacante adivinar una *contraseña* particular a partir de su correspondiente *valor hash*.

El *protocolo de autenticación* descrito anteriormente está entre los más sencillos que alcanzan un nivel adecuado de seguridad, pero en el mismo solo es necesario preocuparse por un número pequeño de amenazas posibles en ese escenario (dejando aparte las posibles amenazas de virus, *key loggers*, etc., contra las que la criptografía no puede hacer absolutamente nada), como son: la fuga de información útil sobre las contraseñas (*password disclosure*), y la posibilidad de que estas puedan ser adivinadas por un adversario externo al sistema (*password guessing*). Notemos que tampoco nos tenemos que preocupar por una posible intercepción, por un atacante, de la información intercambiada entre el usuario y el sistema, pues estamos asumiendo que la autenticación es directa. Veremos cómo, incluso con las medidas tomadas en el protocolo, no se garantiza protección contra la segunda amenaza debido a la debilidad natural que se desprende de la autenticación mediante contraseñas.

De modo general, una contraseña ideal es algo que el *Usuario* conoce, que el *Sistema* puede verificar que el *Usuario* conoce, y que nadie más puede adivinar. Ya hemos visto cómo combinar adecuadamente las dos primeras características deseadas; sin embargo, el último aspecto es, en la práctica, el punto débil, pues para que un valor sea impredecible, este debe ser realmente aleatorio. Además para que no sea sencillo realizar una búsqueda exhaustiva sobre todas las posibles contraseñas, estas deben estar formadas por una cantidad prudente de caracteres. El sentido común nos dice que si una contraseña totalmente aleatoria fuese dada a cada usuario en todos los sistemas, sería un gran problema hoy día, manejar y aprender de memoria tantos valores para casi cualquier persona. Esto iría en contra de la facilidad de uso de los mecanismos de autenticación basados en este método, el cual es uno de los principios fundamentales por el que debemos preocuparnos al diseñar algo de uso popular, como lo es ya el caso de la autenticación basada en contraseñas.

El cifrado unidireccional, no es suficiente para brindar una adecuada protección a un mecanismo de autenticación usando contraseñas, si estas son susceptibles a ser adivinadas. La naturaleza contextualmente humana de las contraseñas estáticas, hace posible el uso de una herramienta bastante efectiva para adivinar las mismas, si estas no son escogidas cuidadosamente. Esta herramienta se denomina *Diccionario*, y consiste una

lista extensa (del orden de millones) de contraseñas comunes o predecibles reunidas por adversarios informáticos, lo que propicia un ataque de búsqueda de contraseñas muy efectivo en la mayoría de los casos, conocido como *ataque de diccionario* (*dictionary attac*). Estos *Diccionarios* por lo general son públicos y se pueden obtener en Internet, haciéndolos más asequibles para aquellos que pretenden violar las medidas de seguridad de los sistemas que se apoyan en las contraseñas como medida de control de acceso.

La fortaleza de una contraseña se puede medir por su nivel de entropía o incertidumbre (Shannon, 1948), y por la probabilidad que tiene esta de aparecer en un *Diccionario* dado. Mientras más entropía tenga la contraseña, o menor sea la probabilidad de que esta aparezca en un *Diccionario*, más segura es la misma.

Por lo planteado anteriormente, es necesario tomar una serie de medidas, y seguir una serie de reglas, a la hora de escoger una contraseña, para poder asumir que esta guarda un grado de seguridad aceptable, las cuales discutiremos a continuación.

Entre los requerimientos para fortalecer las contraseñas están:

- Fijar una cota mínima para el número de caracteres de la contraseña (por lo general de entre 8 y 12).
- Cada contraseña debe contener al menos un carácter de cada clase (mayúsculas, numéricas, alfa-numéricas, ordinarias). Esta medida contribuye a aumentar la entropía de la contraseña escogida.
- Chequear que las contraseñas no estén compuestas por información relacionada a las cuentas como puede ser el identificador de usuario, etc.
- Chequear que las contraseñas no aparezcan en un diccionario común.

Una medida bastante útil y práctica para fortalecer las contraseñas es derivar las mismas de una frese secreta, como puede ser escogiendo como contraseña la cadena de caracteres formada por las primeras (o últimas) letras de cada palabra, y colocando las palabras que representan números por el carácter numérico correspondiente. Por ejemplo “logré dos medallas en tres años y nunca más jugué” puede utilizarse para componer la contraseña “*L2Me3Aynmj”*. De esta forma, los usuarios pueden escoger contraseñas de mayor entropía, y por tanto más fuertes, sin que esto implique para ellos un esfuerzo

inadmisible para recordarlas. Este método es, por el momento, el ideal para que los usuarios escojan una contraseña de libre elección.

Otro riesgo contra el que se quiere brindar la mayor protección posible, es contra aquel en que el atacante logra obtener el archivo del *Sistema*, que contiene la información relacionada con los usuarios y sus contraseñas.

Supongamos que el sistema alberga sólo el *valor hash* de cada contraseña de usuario y no las contraseñas en claro; y que los valores del archivo del sistema son los pares (u_1, h_1) , (u_2, h_2) , (u_3, h_3) ... (u_N, h_N), donde cada $h_i = H(c_i)$, siendo c_i la contraseña del usuario cuyo identificador es u_i. Un tal adversario, que pretende adivinar alguna contraseña del *Sistema*, y tiene en su posesión un *Diccionario*, formado por los valores d_1, d_2, d_3 ... d_M, debe computar el *valor hash* de cada entrada de su diccionario, y luego compararlo con cada uno de los valores del archivo obtenido del sistema, hasta encontrar una coincidencia. El adversario, inteligentemente podría haberse adelantado a este evento, y haber pre-computado una lista de los *valores hash* de su diccionario, teniendo ahora una lista que solo tendría que comparar con los valores robados del sistema, con lo cual el esfuerzo en el momento de la búsqueda de la contraseña sería mínimo. En este caso, la esperanza para un usuario, de que su contraseña no sea descubierta, está en que no aparezca en el diccionario del atacante.

Para que esta forma de proceder no sea factible para el atacante, se puede combinar cada contraseña c_i con un valor aleatorio s_i (llamado *salt*) antes de computar el *valor hash* que se almacenará asociado al identificador u_i, y además mantener una lista aparte de los valores s_i en claro. De este modo, el sistema guardaría los pares $(u_i , H(c_i \parallel s_i))$. Tengamos en cuenta que la *función hash* segura H garantiza que no exista relación estadística útil entre los valores $h_i = H(c_i)$ y $h_i' = H(c_i \parallel s_i)$. Ahora, el atacante deberá conseguir, además, la lista de los valores aleatorios s_i. Luego, para verificar si una contraseña c_i está en su diccionario, debe buscar el valor s_i, calcular un *valor hash* distinto $h_{ij}'' = H(d_j \parallel s_i)$ para cada entrada d_j de su diccionario, y comparar estos valores con valor h_i' del archivo del sistema en busca de alguna coincidencia. Como cada contraseña distinta está combinada con un valor s_i distinto, los *valores hash* ya generados para verificar una contraseña no podrán ser reutilizados en la verificación de las otras.

Los valores aleatorios usados se almacenan en claro, y para adivinar una sola contraseña específica, el atacante no requiere invertir más esfuerzo que en el caso inicial, sin embargo, la búsqueda se realiza ahora en el espacio producto de las contraseñas y los valores aleatorios. Esto implica que para adivinar N contraseñas de usuarios, esta medida supone un aumento del factor de trabajo para el atacante por $N - veces$ más que el esfuerzo inicial, incrementando así la complejidad de un posible ataque basado en diccionario. Vale aclarar que el costo de las operaciones en este caso se mide en función de la cantidad de *hash* computados, como suele ser generalmente.

Esto es un ejemplo claro de cómo los métodos de la seguridad de la información raramente no logran alcanzar una seguridad absoluta, sino más bien pretenden hacer lo más difícil posible el éxito a los adversarios.

(c) Ataques a la autenticación basada en contraseñas estáticas

A continuación presentaremos los principales ataques que suelen tener éxito contra el uso de contraseñas estáticas, lo cual demuestra la debilidad de las mismas y nos impulsará a presentar otras soluciones para dar fortaleza al los procesos de autenticación mediante contraseñas.

1. *Ataque por Búsqueda Exhaustiva de la Contraseña*: El ataque menos elaborado para este escenario, es aquel mediante cual el atacante escoge contraseñas al azar y trata de verificar si ha acertado. Esta verificación se puede basar en dos cuestiones distintas: una es de forma directa con el sistema, y en tiempo real (*online verification*); y la otra es a partir de una copia fraudulenta, del archivo de verificación de las contraseñas del sistema.

   En el primer caso, el adversario deberá realizar repetidos intentos de autenticarse con el sistema y sabrá si ha tenido éxito o no basado en la respuesta del sistema. Esto puede ser contrarrestado, limitando los intentos fallidos de autenticación que puede realizar cada supuesto usuario del sistema. Sin embargo, esta medida que parece efectiva a simple vista, puede ser usada por un atacante para limitar o

impedir a un usuario genuino del sistema, el acceso al mismo durante un tiempo prolongado. Lo último es conocido como *Ataque de Negación de Servicios*. Este inconveniente no suele ser sencillo de solucionar del todo y las posibles medidas para solucionar esta problemática particular pertenecen más bien al diseño de sistemas para la autenticación, por lo que está fuera del alcance de los objetivos de nuestro trabajo, y lo dejaremos sin más explicación.

El segundo caso es aquel en el que el adversario posee una copia del archivo del *Sistema,* y lo utiliza para verificar la validez de las contraseñas de los usuarios. Supongamos que este archivo está formado por la lista de pares $(u_1, f(p_1)), (u_2, f(p_2)), (u_3, f(p_3)) \dots, (u_N, f(p_m))$ donde cada p_i es la contraseña del usuario u_i . Con ello, el atacante podrá por su propia cuenta, sin depender del sistema, generar valores aleatorios c_j, dentro del espacio de las contraseñas admisibles, y aplicar a cada valor la función de verificación $f(\cdot)$, para realizar así una búsqueda exhaustiva comparando con los valores del archivo en busca de coincidencias. El atacante podrá además, hacer uso de toda la potencia computacional de la que pueda disponer para lograr sus objetivos. Suponiendo que el espacio de contraseñas admisibles es de N elementos, en las dos posibilidades contempladas hasta ahora para los verificadores de contraseñas, la cantidad de *operaciones hash* esperadas por parte del adversario antes de que este logre encontrar una coincidencia es $\frac{N}{2}$ (basado en la teoría de las probabilidades). Aclaremos que el tiempo requerido para realizar la comparación de un valor contra todos los valores del archivo de los verificadores, es por lo general despreciado y solo se cuenta la cantidad de operaciones criptográficas, como lo son las *operaciones hash*.

Visto todo lo anterior, podemos llegar a la conclusión de que en ambos casos la mejor medida contra este ataque de fuerza bruta es lograr que el espacio de contraseñas tenga un tamaño suficientemente grande como para que no le sea factible a un atacante realizar esta cantidad de operaciones.

2. *Password Guessing y ataques de diccionario*: Para mejorar la probabilidad de éxito esperada, en vez de realizar una búsqueda exhaustiva “a ciegas” sobre todo

el espacio de contraseñas, el adversario del sistema puede realizar una búsqueda en orden decreciente de la probabilidad esperada de los elementos. Esto es posible debido a que la mayoría de los usuarios escoge sus contraseñas dentro de un pequeño subconjunto del espacio de contraseñas; y algunos de ellos escogen contraseñas demasiado comunes y predecibles al asociarlas a objetos comunes de uso general, entre otras cosas; lo cual ha dado lugar a los *Diccionarios*, que ya hemos mencionado. En estos *Diccionarios* aparecerán las palabras que más probabilidad tienen, de acuerdo a la práctica, de ser elegidas como contraseñas por un usuario común, por lo general ordenadas descendentemente de acuerdo a su probabilidad. Las contraseñas con poca entropía o que aparecen en algún diccionario son contraseñas débiles y pueden ser fácilmente adivinadas por un adversario sin invertir mucho tiempo y recursos. En este ataque, el adversario procederá de forma similar al ataque anterior, con la importante diferencia de que se apoyará en un *Diccionario*, realizando una búsqueda inteligente de las contraseñas, y no "a ciegas", con lo cual se incrementan notablemente sus probabilidades de éxito.

Una medida de lo efectivo que puede ser un ataque usando un diccionario, lo dan los resultados de estudios, que indican que un gran porcentaje de las contraseñas escogidas de por los usuarios pueden ser encontradas en diccionarios de 1500 palabras aún cuando un diccionario común de 2500 palabras sólo representa una pequeña fracción del espacio usual de contraseñas.

3. *Eavesdropping Attack*: Este es el ataque más simple entre los ataques en que el adversario se involucra de forma pasiva. El mismo consiste en que el atacante monitorea las comunicaciones entre dos o más entes, interceptado los mensajes enviados durante estas; y trata de extraer alguna información útil de estas comunicaciones, que le permita burlar la seguridad de algún modo y lograr sus objetivos. En el caso de las contraseñas estáticas, el atacante podría aprovecharse del tipo de información que sea enviada para la corroboración de la contraseña, y usarla para autenticarse con el sistema en próximos intentos. La principal medida

para evitar el la efectividad del ataque *eavesdropping* es el cifrado de los mensajes que puedan comprometer la seguridad.

4. *Replay Attack*: Otra de las vulnerabilidades de los esquemas de autenticación que usan contraseñas fijas, y por tanto reusables, es que si estas son transmitidas por un canal inseguro, y son recolectadas por un adversario en modo *eavesdropping*, este podría usar los mensajes recolectados de comunicaciones anteriores para autenticarse con el *Sistema*, suplantado así a un usuario genuino. En el caso de las contraseñas estáticas este tipo de ataque será efectivo si no se toman ciertas medidas que van más allá de las ilustradas hasta este momento.

De este modo, los esquemas de autenticación mediante contraseñas estáticas son adecuados sólo cuando se envía la información por un canal seguro de comunicaciones, y no son adecuadas para el caso en que las comunicaciones son transmitidas dentro de una red abierta. En algunos casos, además de proteger la contraseña enviada, se debe proteger incluso la respuesta del sistema (aceptado/rechazado). La mejor forma de protección contra los ataques mencionados, es hacer variable o dinámica a la información de autenticación enviada entre las partes.

Los dos últimos ataques entran dentro de la clase de ataques generales a los protocolos de autenticación. Cuyo estudio comenzaremos en el próximo capítulo.

Hasta el momento creemos que con el presente recorrido teórico es suficiente para entender la esencia y las propiedades fundamentales que hacen de gran utilidad las herramientas presentadas en este capítulo para el ámbito que nos concierne. En lo que sigue, veremos las formas más adecuadas de usar y combinar estas herramientas para obtener una seguridad aceptable en la autenticación robusta, y para alcanzar los objetivos del presente trabajo.

CAPÍTULO 2: ASPECTOS METODOLÓGICOS

En el presente capítulo nos dedicaremos a sentar las bases y delinear algunas pautas para el análisis y el diseño de los protocolos de autenticación robusta. Además de ello, estudiaremos algunos métodos importantes para fortalecer la autenticación usando secretos compartidos, e ilustraremos el proceso de construcción de un protocolo más seguro a partir de un protocolo básico. También delinearemos el marco de los tipos de protocolos de autenticación con los que trataremos en este trabajo, y presentaremos las contraseñas de un solo uso (OTPs), las cuales son una de las herramientas más importantes para el desarrollo de esta tesis. Con lo presentado en el capítulo anterior y lo que sigue en el presente capítulo quedarán sentadas las bases necesarias para la comprensión y el análisis de protocolos de autenticación robusta bastante complejos, como es el caso del protocolo de Vaidya et al (2011) que se verá en el capítulo siguiente.

2.1 Bases para el análisis y diseño de los protocolos de autenticación de entidades

En sentido general los protocolos son el grupo de reglas determinadas que se deben seguir en una situación o interacción determinada.

Definición 2.1: Para el contexto que nos ocupa, un *protocolo de autenticación* es el grupo de reglas de comunicación, que se deben seguir en un *Sistema* de comunicaciones, para que un grupo de entes (llamados *Verificadores*) logren verificar la validez de la identidad de un grupo de entes *Solicitantes*. Estos dos grupos de entes pueden hacer uso de otro grupo de entes intermediarios, en los cuales depositarán su confianza.

El objetivo fundamental de un Protocolo de Autenticación es que todo proceso de verificación de identidad se lleve a cabo correctamente y sin fraude de ninguna de las partes que intervienen. Se puede decir que un protocolo es seguro si cumple con los objetivos para los que ha sido diseñado en un marco dado. En las próximas secciones

daremos una idea lo mejor posible del procedimiento a seguir para diseñar un buen protocolo de autenticación, que pretenda brindar seguridad en ciertos aspectos determinados.

Aunque podría parecer que el tema de los *protocolos de autenticación* es una recurrencia al mismo tema de la *autenticación* en sí, que hemos estado tratando, debemos estar claros de que no es así. Al hablar de estos protocolos debemos preocuparnos por una serie de cuestiones con las que no es necesario lidiar cuando se habla de un mecanismo de autenticación, como son los aspectos de seguridad que tienen que ver con los mensajes que deben ser enviados para que se pueda llevar a cabo la autenticación. Por otra parte, para poder concentrarnos en este tipo de aspectos, asumiremos que los tipos de cifrados que se utilicen son seguros. Así, en el ámbito de los protocolos de autenticación, aparecen toda una serie de ataques que tienen que ver con los mensajes que son intercambiados en la ejecución del protocolo, y no relacionados directamente con los métodos criptográficos usados. De modo que, no nos ocuparemos de los ataques relacionados intrínsecamente con la criptografía utilizada en los protocolos, y asumiremos que las operaciones criptográficas usadas son seguras.

Los protocolos en general son muy sutiles. Usualmente, un cambio ligero y aparentemente insignificante, en un protocolo puede hacer una diferencia significativa en el mismo. En particular, los protocolos de seguridad están entre los más delicados, ya que los atacantes pueden intervenir en el proceso en una gran variedad de formas. Como muestra del reto que supone el diseño de este tipo de protocolos, mencionemos que algunos protocolos de seguridad bien conocidos como los son: IPSec, WEP, tienen serias fallas de seguridad. Incluso si el protocolo no tiene fallas, una implementación particular del mismo podría tenerlas.

A la hora de diseñar un protocolo de autenticación en general, es necesario tener en cuenta una serie de aspectos fundamentales, la omisión de alguno de los cuales podría conducir a serias fallas de seguridad o funcionalidad; a continuación mencionamos los que consideramos de mayor importancia para el marco de este trabajo, para lo cual nos hemos basado en (Menezes et al., 1997) y (Colin & Anish, 2003):

1. *Reciprocidad en la identificación*: salvo en casos excepcionales, cada una de las partes debe corroborar su identidad, proporcionándose una autenticación mutua.
2. *Identificación explícita*: Si la identidad de un ente principal es fundamental para el sentido de un mensaje dentro del protocolo, entonces el identificador de este ente debe ser incluido explícitamente en el mensaje.
3. *Propiedades de los cifrados*: Es importante estar claros de las razones por las que se usa cada cifrado en el protocolo y qué propiedades debe proporcionar.
4. *Envejecimiento de claves*: Se debe tener en cuenta el tiempo o período por el cual pueden ser usados los secretos de largo y corto término antes de que puedan ser comprometidos.
5. *Eficiencia computacional*: El número de operaciones necesarias para ejecutar el protocolo debe ser aceptable con respecto al entorno donde este es puesto en práctica.
6. *Eficiencia en las comunicaciones*: En implementaciones particulares es importante tener en cuenta el número de mensajes intercambiados en el protocolo y el ancho de banda requerido, el retardo, entre otras cosas.
7. *Almacenamiento de secretos*: Es importante tener en cuenta qué tipo de información almacenará cada ente sobre los secretos que no le son propios, y en algunos casos la forma y ubicación de esta información almacenada.
8. *Arquitectura de comunicaciones*: La arquitectura de comunicaciones (en los casos en que lo requiera) debe ser definida, así como el papel que asume cada ente que participe en el protocolo.
9. *Actuación de cada principal*: La forma de actuar de cada ente principal, ante cada intercambio de información en el que se ve involucrado, debe ser incluida explícitamente en la descripción del protocolo.
10. *Amigable para el usuario*: si en el protocolo un *Usuario* debe tomar acciones, entonces se tratará de que este lo haga en la forma más amigable posible, en su interacción con el resto de los entes. Como consecuencia el protocolo debe ser tan fácil de usar como se pueda.

11. *Intervención de terceros*: La intervención de una tercera parte de confianza, en tiempo real, puede ser necesaria en algunas ocasiones, ya sea para distribuir claves comunes a ambas partes principales o para la distribución de certificados públicos validados por una Autoridad Certificadora. Este tipo de intervención no se debe utilizar a no ser que existan buenas razones para ello; y es importante en estos casos, hacer explícito el nivel de confianza que se le da a esta tercera parte.
12. *Identificación de sesiones*: En el intercambio de información, los entes principales deben poder distinguir si dos mensajes dados son de distintas ejecuciones del protocolo o no.

Estas reglas se derivan de la observación y corrección de errores comúnmente cometidos en el diseño de protocolos de autenticación. Aun así, el cumplimiento de las mismas no garantiza del todo el diseño de un buen protocolo; otros elementos deben ser tenidos en cuenta, los cuales iremos abordando poco a poco.

En lo que sigue nos dedicaremos a estudiar algunas pautas para escalar la seguridad de los protocolos de autenticación de entidades que se basan en, al menos, un secreto compartido (lo cual se ajusta a la clase de autenticación mediante “algo conocido”); con ello preparamos las bases para adentrarnos luego en el estudio de aquellos que se basan en contraseñas. Teniendo en cuenta que en la gran mayoría de los casos la información intercambiada en la ejecución de los protocolos atraviesa por redes inseguras, consideraremos en todos los casos subsiguientes que los canales de comunicación, entre los entes que intervienen, son inseguros, a no ser que se aclare explícitamente el uso de algún vínculo seguro. Además, por razones de eficiencia computacional, solo estudiaremos protocolos que usen criptografía simétrica u operaciones ligeras, como lo son las operaciones *Hash* y *XOR*. Estos métodos requieren, por lo general, más ingenio y cuidado, por lo que pueden conducir a protocolos bastante complejos. La mayoría de la bibliografía sobre protocolos de seguridad y autenticación está dedicada al intercambio seguro de claves, por lo que muchas de las ideas planteadas en este trabajo han sido adaptadas, y no directamente extraídas de ese tipo de esquemas.

El conocimiento sobre algoritmos criptográficos y las propiedades de los mismos es importante para el correcto diseño de implementaciones particulares de un protocolo. Sin

embargo, los detalles de qué algoritmos deben ser usados exactamente dentro de una clase específica pueden considerarse irrelevantes a la hora de diseñar y analizar un protocolo de autenticación genérico. Como consecuencia, también los ataques considerados serán independientes de algoritmos criptográficos en particular. De cualquier forma, para el diseño de estos protocolos, siempre se debe señalar la existencia de algoritmos criptográficos que cumplan con los requerimientos de seguridad que se van a asumir.

En la bibliografía es común referirse a los entes que originalmente deben participar en un protocolo, como *entes principales*; término que estaremos usando con frecuencia a partir de ahora. En todo momento asumiremos que todos los *entes principales* deben actuar honestamente, de forma que descartaremos la posibilidad de que un atacante pueda corromper a un *ente principal* y aprovecharse de esto para tener éxito. De este modo no estaremos interesados en ataques internos (desde adentro del mismo *Sistema*) contra los protocolo que analizaremos.

En la descripción de nuestros primeros protocolos, por ser bastante simples y generales, sólo incluiremos los intercambios de mensajes dentro de una ejecución exitosa del mismo, utilizando una notación compacta y convencional, sin incluir de forma precisa la forma de actuar de los entes que intervienen en el proceso. Esto suele ser de omisión común en la gran mayoría de la bibliografía sobre protocolos de autenticación; sin embargo, reconocemos que este tipo de información es importante para el planteamiento de ciertas propiedades funcionales de los protocolos, por lo que la iremos incluyendo a medida que avancemos hacia protocolos más complejos y aterrizados en escenarios específicos. Señalemos además que, de acuerdo con lo asumido en el párrafo anterior, ningún atacante podrá intervenir en el protocolo como un ente principal legítimo.

2.2 Potencialidades del atacante y bases para la seguridad

Aunque por lo general no se intercambian muchos mensajes en una instancia de un protocolo, existe una enorme cantidad de formas y combinaciones en las que un atacante puede intervenir; sin embargo, la gran mayoría de estas son trivialmente insatisfactorias para el mismo. La seguridad del protocolo aumenta a medida que la cantidad de posibles ataques efectivos se reduce. Por lo tanto, una manera de alcanzar la seguridad requerida en un protocolo de autenticación es por aproximaciones progresivas, mediante la reducción sucesiva de posibles ataques efectivos contra este en el escenario en cuestión.

Para poder prever las diferentes formas en que un atacante puede intervenir en nuestro protocolo, debemos definir las posibles acciones que este podrá realizar de modo general. En lo que sigue, iremos escalando la seguridad de los protocolos que estudiaremos, teniendo en cuenta las potencialidades de los atacantes, hasta obtener un máximo de protección deseado.

A partir de ahora, un atacante o intruso será denotado de modo general por T, aludiendo a lo más común en la bibliografía anglosajona, en la cual, se suelen referir al intruso (*Intruder*) como *Trudy*. En primara instancia, asumiremos que el atacante T podrá realizar las siguientes acciones, a las cuales se irán agregando otras posteriormente:

1. Monitorear regularmente, las ejecuciones del protocolo entre los entes que intervienen en el mismo e interceptar sus mensajes, e incluso almacenarlos.
2. Invitar (solicitar), a los entes que corresponda, a establecer nuevas sesiones de autenticación o ejecuciones del protocolo.
3. Establecer tantas sesiones de autenticación paralelas como desee, ejecutando así varias instancias del protocolo al mismo tiempo, que él podrá controlar.
4. Establecer comunicaciones con cualquiera de los entes usando mensajes de comunicaciones anteriores del protocolo, intentando suplantar la identidad de alguno.
5. Modificar mensajes espiados y enviar mensajes arbitrarios a todos los entes.

Por otra parte, con el objetivo de ayudarnos a evaluar la seguridad de nuestros protocolos, vamos a definir algunos objetivos generales fundamentales para los mismos (Colin & Anish, 2003):

1. Cada entidad debe asegurarse de que la entidad par con la que pretende comunicarse ha participado realmente en la sesión actual del protocolo.
2. Si durante el protocolo se establece algún secreto de corto término, cada parte debe tener conocimiento de cuáles otros entes están en posesión de este secreto cuando haya sido establecido.
3. Cada ente principal debe tener certeza de que aquel con quien pretende comunicarse en un momento dado, lo reconoce como su ente par en la comunicación.
4. Si una ejecución del protocolo ha sido completada exitosamente, entonces ningún ente que haya intervenido activamente habrá cometido fraude durante el mismo.

La definición de los, antes mencionados, objetivos generales para la autenticación de entidades, es un asunto un tanto polémico, como puede ser corroborado por el estudio de (Colin & Anish, 2003). La forma en que se materializa uno u otro objetivo, puede depender del marco de aplicación del protocolo y las intenciones de sus entes principales, para el cumplimiento de las cuales el protocolo ha sido diseñado. Esto indica que también es importante considerar objetivos particulares para cada tipo de marco de aplicación del protocolo, los cuales pueden ser señalados explícitamente solo cuando el caso lo requiera debido a la complejidad y dificultad de análisis.

La finalidad práctica de los objetivos es ayudar a determinar cuándo un ataque al protocolo es válido o no, lo cual conduce a una definición formal de lo que se entiende por un protocolo seguro.

Definición 2.2: Un protocolo de autenticación de entidades puede ser considerado seguro, si no es posible realizar un ataque que viole alguno de los objetivos anteriormente planteados.

Por consiguiente, daremos por *válido* cualquier *ataque* que impida el cumplimiento de alguno del estos objetivos generales en un protocolo dado.

2.3 Mecanismos de protección contra ataques básicos en la autenticación mutua

El marco general más simple para un protocolo de autenticación de entes, involucra dos *principales*: un ente *solicitante* A y un ente *verificador* B. El ente *solicitante* debe presentar alguna evidencia al ente *verificador* de que es quién dice ser. El ente verificador conoce la identidad del ente genuino A, o la asume de ante mano. Durante el proceso, B debe corroborar que el solicitante es precisamente A y no otro distinto que dice serlo; esto debe ocurrir en tiempo real. Lo descrito es un proceso de autenticación unidireccional entre dos entidades. Veamos ahora como construir un protocolo concreto para lograrlo, de forma aceptable.

Para nuestros primeros protocolos bipartitos designaremos por A y B, a los *participantes principales*. Entre estos dos entes no habrá distinción específica, por lo tanto cualquiera de ellos podrá iniciar el protocolo, y sus papeles podrán ser intercambiados. Asumiremos que ambos comparten un secreto común K el cual ha sido establecido de forma segura, tal que ambos concuerdan en que es "bueno" para ser usado con fines criptográficos. Asumiendo que este secreto es sólo conocido por ambas partes, el mismo puede ser usado como base para la autenticación mutua. A la vez, este secreto puede ser visto como una clave segura de largo término. Inicialmente, descartaremos la posibilidad de un robo de secretos por parte un atacante, cuestión con la que comenzaremos a lidiar cuando tratemos con protocolos más complejos.

Bajo las condiciones planteadas, digamos que A necesita autenticarse con B, lo cual hará mediante el uso del secreto compartido K. En busca de una primera aproximación de nuestro protocolo seguro, debemos tener en cuenta que no es correcto que A envíe este secreto en claro, ya que podría ser interceptado por un atacante T en modo *eavesdropping*, y luego usado para autenticarse como B; lo cual se entiende como una *suplantación de identidad*.

Definición 2.3: Un *ataque de suplantación de identidad* (*impersonation attack*) es aquel en el cual un ente asume la identidad de otro y logra que un tercero crea que es quien dice ser. En otras palabras, un ente malicioso, portando algún tipo de credenciales, que no le son propias, logra ser autenticado como un ente distinto. Se puede decir que el

ente malicioso usa una identidad falsa para enmascararse, y pasar como algo que realmente no es. Por esto, a este tipo de ataque, también se le suele llamar *masqueradin attack.*

Una medida para no dejar al descubierto el secreto compartido sería que A envíe el *valor hash* del mismo a B. Aún así, la información que usa A para autenticarse es estática, por lo que esta medida no evitaría un *replay-attack*, que le permitiría a T ser autenticado como A incluso sin conocer el secreto K.

$$A \longrightarrow B : A, H(K)$$

Para brindar protección contra los ataques de tipo *replay-attack*, se puede usar lo que se llama un mecanismo de *desafío-respuesta* (*challenge-response*). La idea general de un esquema *desafío-respuesta*, es que si un ente desea recibir un mensaje de su ente par, del cual necesita asegurarse de que es un mensaje nuevo, este usará un parámetro variable con el tiempo, y no repetitivo, para establecer un "reto" que supuestamente sólo su ente par genuino podría responder correctamente. Aquí la propiedad de no repetitividad es importante para asegurar que el valor no haya sido usado anteriormente, y el reto actual no haya sido respondido en un momento anterior; con lo cual, un atacante podría tener la respuesta actual. Para lograr esto, es importante que el ente verificador controle el parámetro variable en el tiempo usado. Al recibir la respuesta correcta, dado que la misma estará asociada a un valor nuevo del parámetro, será considerada como una respuesta actual, y por tanto, descartada la posibilidad de que sea una repetición de una respuesta anterior.

Veremos primero un esquema general de *desafío-respuesta* para este caso, seguido del cual iremos desarrollando implementaciones particulares en orden de ir incorporando aspectos de seguridad requeridos.

Cuadro 1: Esquema general de desafío-respuesta.

$paso\ 1.\ A \longrightarrow B : "Soy\ A"$
$paso\ 2.\ B \longrightarrow A :$ "$Nonce$" (*number used once*)
$paso\ 3.\ A \longrightarrow B : "Algo\ que\ solo\ puede\ conocer\ A\ y\ que\ B\ puede\ verificar"$

El esquema general de desafío respuesta consta de los tres pasos señalados en la Fig. 1; donde el ente que inicia el protocolo (ente solicitante) se identifica con su ente par (ente verificador), luego de lo cual su ente par le envía a este un valor usualmente denominado *nonce* (*number used once*) distinto para cada ejecución protocolo, y finalmente este valor es usado para generar una respuesta dirigida a su ente par. Esta respuesta debe ser tal que ningún otro ente haya podido remitir, y cuya validez pueda ser comprobada por el ente verificador.

En el protocolo que sigue a continuación, B le enviará un número aleatorio al ente A, que servirá como desafío, el cual será entonces utilizado por A para componer una respuesta que sólo él podía producir y solo B puede verificar si es correcta. De de este modo, B enviará un desafío distinto cada vez al ente A, y este desafío será necesario para obtener la respuesta adecuada cada vez. Con ello B podrá distinguir la respuesta actual, de respuestas anteriores.

Cuadro 2: Protocolo según el esquema general de desafío-respuesta.

1. $A \longrightarrow B : A$
2. $B \longrightarrow A : R$
3. $A \longrightarrow B : f(R', K)$

Para que un esquema que se ajuste al protocolo que se muestra en el Cuadro 2 sea efectivo, es importante que se cumplan las siguientes condiciones:

- El desafío debe ser criptográficamente ligado al secreto que constituye la base de la autenticación, para evitar que un atacante pueda calcular el valor correcto de la respuesta actual $f(R, K)$, a partir de las respuestas anteriores.
- Los desafíos no se deben repetir en distintas ejecuciones del protocolo pues, de lo contrario, sería posible un *replay attack* si el atacante ha almacenado alguna respuesta anterior correspondiente al desafío repetido.

- Para cumplir con el primer punto, y garantizar la seguridad del secreto K, se debe exigir que la función criptográfica f posea las propiedades de confidencialidad y no-maleabilidad (ver propiedades de la 4 a la 6 de las *funciones hash* en el Capítulo 1).
- Por último la función criptográfica f debe ser libre de colisiones o fuertemente resistente a las mismas.

En el último paso del protocolo que muestra la Cuadro 2, B debe calcular $f(R, K)$ usando el valor R que tiene almacenado, y compararlo con el valor recibido $f(R', K)$. Si estos coinciden, entonces A será autenticado positivamente basado en el hecho de que sólo quien conozca el secreto compartido K, y haya recibido el valor aleatorio R, podrá haber generado el valor correcto $f(R, K)$.

Bajo las condiciones exigidas a la función criptográfica f, está bien puede ser una *función hash segura*, un *cifrado simétrico con la clave K* o un *código MAC* del desafío *R* usando la clave K. O sea:

$$f(R,K) = \begin{cases} H(R,K) \\ \mathcal{E}(R,K) = \mathcal{E}_k\{R\} \\ MAC(R,K) \end{cases} \quad (1)$$

Ahora notemos que, a pesar de que el protocolo de la Cuadro 2 parece ser adecuado, no es aceptable para nuestros objetivos generales, ya que la parte solicitante A no recibe ninguna seguridad de que su entidad par, con la cual pretende autenticarse, ha participado en el proceso. Un atacante puede aprovechar esto para interponerse en las comunicaciones y escoger los desafíos R_i a su antojo (como se muestra en el ataque del Cuadro 3); logrando así recibir de parte de A suficiente información cifrada $f(R_i, K)$ como para realizar un criptoanálisis, y obtener K.

Esta problemática conduce a la necesidad de autenticación mutua. Por el momento nos centraremos en el caso en que f es una *función hash segura*, ya que es el que resulta más eficiente computacionalmente, asunto en el que nos interesa enfatizar.

Cuadro 3: Ataque a un mecanismo básico de desafío-respuesta.

1. $A \longrightarrow T : A$
2. $T \longrightarrow A : R_i$
3. $A \longrightarrow T : H(R_i, K)$

La autenticación unidireccional es aceptable para casos de autenticación directa, como aquel en que un usuario se autentica con un ordenador conocido, en una red de confianza, para acceder a este. Pero en la mayoría de los casos se requiere de autenticación mutua, para evitar que alguno de los entes participantes en el protocolo establezca secretos, o intercambie información comprometedora para la seguridad, con un ente cuya identidad no ha corroborado, y que por tanto, podría ser una atacante. Estos casos se dan, sobre todo, cuando la información atraviesa en algún momento por un canal de comunicaciones inseguro, lo cual es así casi siempre.

En los subsiguientes análisis queremos evitar la criptografía reversible y utilizar una *función hash* siempre que esto sea posible antes que usar *cifrado de clave simétrica*. Nuestro próximo paso será tratar de introducir autenticación mutua en el último esquema. Puede parecer obvio que el mismo procedimiento que es usado para que A pruebe su identidad a B, podría ser usado en la otra dirección para autenticar a B; sin embargo, en los protocolos de seguridad, las soluciones obvias suelen ser frecuentemente erradas.

En primera instancia, vamos a tratar de enviar el mismo método de desafío respuesta en sentido inverso para lograr nuestro objetivo. Una forma de intentarlo sería mediante el siguiente protocolo.

Cuadro 4: Protocolo simétrico de autenticación mutua mediante desafío-respuesta.

1. $A \longrightarrow B : A , R_A$
2. $B \longrightarrow A : H(R_A, K) , R_B$
3. $A \longrightarrow B : H(R_B, K)$

En este caso, A inicia el protocolo enviando un desafío a B junto con su identificador. B asocia criptográficamente el secreto compartido K con el desafío que ha recibido, para autenticarse con A, y añade un desafío propio a la información enviada. Este desafío será utilizado por A en sentido inverso para ser autenticado por B, utilizando el mimo tipo de mensaje. Ambos entes deben confiar el uno en el otro en que cada uno sólo producirá valores de desafío irrepetibles en su propia secuencia. Por otra parte, en este protocolo, B deberá verificar que el desafío que escoja sea distinto del recibido anteriormente dentro de la misma ejecución del protocolo (o sea $R_B \neq R_A$); de lo contrario el protocolo será trivialmente susceptible a un *replay attack*.

Sin embargo, estas precauciones no son suficientes. Un supuesto bastante común es que el atacante puede establecer distintas sesiones paralelas del protocolo e involucrarse en estas de uno u otro modo, tal y como lo hemos planteado en la potencialidad del atacante número 3) del acápite 2.2. Pero en este protocolo, a pesar de que hemos logrado introducir un mecanismo para distinguir entre mensajes de las sesiones actuales y mensajes de las sesiones anteriores, no hay forma de distinguir entre mensajes actuales de sesiones paralelas. Por otra parte, la autenticación de ambos entes se realiza de la misma forma y por medio del mismo tipo de mensajes; esto hace que el protocolo sea simétrico lo cual es peligroso.

Aunque en muchos ámbitos de las ciencias la simetría es un aspecto elegante y deseado, en este caso solo trae problemas. Veremos cómo es una mala idea tener a dos partes en un protocolo haciendo lo mismo, ya que esto deja abierta una oportunidad para un *replay attack*.

Denotaremos por T_X a un atacante T que finge ser el ente X (en este caso $X = A\ o\ B$) en una instancia del protocolo.

Una de las vulnerabilidades del protocolo del Cuadro 4 proviene de la posibilidad de intercambiar los roles de A y B, dando lugar a que cualquiera de ellos pueda iniciar una sesión paralela; lo cual, unido a lo anteriormente planteado, conduce a un tipo de ataque llamado *reflection attack*, que puede implementarse como se muestra a continuación. El atacante T puede proceder de la siguiente forma:

Cuadro 5: Ataque: Reflection attack

1. $A \longrightarrow T_B : A , R_A$
$1'.\ T_B \longrightarrow A : B , R_A$ [*Inicio de una nueva sesión paralela*]
$2'.\ A \longrightarrow T_B : H(R_A, K) , R'_A$
$2.\ T_B \longrightarrow A : H(R_A, K) , R'_A$
$3.\ A \longrightarrow T_B : H(R'_A, K)$
$3'.\ T_B \longrightarrow A : H(R'_A, K)$

Como muestra el esquema del Ataque representado en la Cuadro 5, el atacante T se interpone en la primera ejecución del protocolo asumiendo el papel de B, este solo interactúa con A, quien es el iniciador de la primera instancia del protocolo. Como en este caso los papeles de A y B pueden ser cambiados, el atacante aprovecha esto para iniciar otra sesión del protocolo en el sentido contrario con el mismo A. Esta nueva sesión paralela es usada por el atacante para obtener la información que necesita, para ser autenticado por A en la sesión inicial activa, suplantando la identidad de B. Los desafíos que le son enviados al atacante en cada una de las sesiones son regresados al ente A en la otra sesión paralela. Como A siempre responderá correctamente, el atacante puede usar las respuestas de vuelta contra el mismo A. De este modo, el atacante T logra completar exitosamente ambas ejecuciones del protocolo sin que A note que su ente par B, ni siquiera ha intervenido en el protocolo en ningún momento.

Este ataque ingenioso es llamado *reflection attack*, ya que los mensajes enviados por la parte genuina son regresados al mismo para obtener respuestas correctas, pudiéndose decir que el atacante funciona como un "reflector" en cierta forma.

Una medida para evitar el *reflection attack*, consiste en incluir, en la respuesta, el identificador del emisor o el destinatario, criptográficamente ligado al valor de desafío y al secreto K. Con ello, el atacante no podría completar exitosamente ninguna de las sesiones del protocolo sin interactuar con la otra parte, ya que no conoce el secreto compartido K. Así, por ejemplo, en el primer caso, B solo produciría respuestas ligadas al identificador B y esperaría respuestas ligadas al identificador A. De este modo, B al recibir una respuesta asociada con el identificador de A, reconocería al menos que este

mensaje no es una repetición de un mensaje enviado por si mismo. El caso inverso también es igualmente usado, con la ventaja de que cada ente recibe una señal de que aquel con el que se comunica lo reconoce como su ente par en la ejecución del protocolo (objetivo 3). Veamos una variante en la que se incluyen los identificadores.

Cuadro 6: Protocolo de autenticación mutua, mediante desafío-respuesta, usando identificadores implícitos.

1. $A \longrightarrow B : A , R_A$
2. $B \longrightarrow A : H(B, R_A, K) , R_B$
3. $A \longrightarrow B : H(A, R_B, K)$

Aun así, el atacante puede interponerse entre A y B, y auxiliarse de uno de ellos para lograr ser autenticado positivamente por el otro, llevando a cabo una exitosa *suplantación de identidad*.

Cuadro 7: Ataque: Interleaving attack.

1. $T_A \longrightarrow B : A , R_T$
2. $B \longrightarrow T_A : H(B, R_T, K) , R_B$

$1'$. $T_B \longrightarrow A : B , R_B$

$2'$. $A \longrightarrow T_B : H(A, R_B, K) , R_A$

3. $T_A \longrightarrow B : H(A, R_B, K)$

En este caso, el atacante T es el iniciador de ambas sesiones paralelas del protocolo. Solo que estas vez, la primera sesión la establece con B, y la segunda con A. En esta segunda sesión el atacante usa el desafío enviado por B en la primara instancia del protocolo, para obtener la respuesta correcta de parte de A y usarla contra B en la en la primera sesión activa. Aunque en esta ocasión la segunda instancia del protocolo no puede ser completada exitosamente por el atacante, este sí lo logra en la ejecución inicial del protocolo, sin que B sospeche que A ni siquiera ha intervenido en la primera sesión; violándose así el objetivo 1 para la autenticación de entidades.

En estos casos se dice que la parte consultada por el atacante actúa como un *Oráculo*, ya que le brinda una guía al adversario para continuar exitosamente una sesión paralela activa, vulnerando la seguridad del protocolo. Notemos que tanto en el ataque anterior como en el actual, A es usado como *Oráculo* por el atacante T. Este ataque desarrollado, en el cual el adversario interactúa con ambas partes genuinas del protocolo, en sesiones paralelas, usando uno de ellos como *Oráculo* para completar con éxito una de las ejecuciones del protocolo, se denomina *interleaving attack*.

Lo visto hasta ahora, es una muestra de lo inseguro que resulta que la información intercambiada, entre las partes dos a dos de un protocolo, sea simétrica, así como que los entes principales cumplan la misma función en el modelo de comunicaciones. Existe una herramienta formal para evitar este tipo de ataques, denominada cotejado de conversaciones -*matching conversations*- (Colin & Anish, 2003). Esta plantea un modelo para asegurar que el atacante no pueda interactuar exitosamente entre distintas sesiones del mismo protocolo. Para mantener el lenguaje simple y no entrar en formalizaciones que puedan quitar claridad a nuestras ideas, mencionemos que la base de este método está en romper la simetría del protocolo y buscar una forma de vincular lógicamente los mensajes consecutivos de una ejecución del protocolo, de forma tal que estos sólo puedan ser asociados a una sesión del mismo. En la mayoría de los casos con asegurarse de etas condiciones basta para evitar el ataque.

El siguiente protocolo general, propuesto por Bird et al. sugiere usar dos tipos de mensajes distintos para cada una de las direcciones en las cuales es realizada la autenticación, con el objetivo de cumplir con el cotejado de conversaciones -*matching conversations*- (Bird et al., 1993).

Cuadro 8: Protocolo canónico de Bird et al.

1. $A \longrightarrow B : N_A$
2. $B \longrightarrow A : N_B , u(K_{AB}, N_A, \dots)$
3. $A \longrightarrow B : v(K_{AB}, N_B, \dots)$

Aquí N_A y N_B representan valores de *nonce* y las funciones u y v han de ser distintas. No obstante el protocolo puede seguir siendo vulnerable aún siendo $u \neq v$, si no se escogen adecuadamente estas dos funciones.

Un ejemplo más concreto en el cual se previenen todos los ataques anteriores cumpliendo con el macheo de conversaciones, se puede obtener mediante una adaptación del protocolo 2PKDP propuesto por Janson y Tsudik en (Colin & Anish, 2003), con el objetivo de proveer autenticación muta entre dos entidades:

Cuadro 9: Adaptación del protocolo Janson-Tsudik 2PKDP para autenticación de entidades

1. $A \longrightarrow B : A, N_A$
2. $B \longrightarrow A : AUTH, MASC \oplus N_B$
3. $A \longrightarrow B : MAC_{K_{AB}}(N_A, N_B, A)$

Donde $AUTH = MAC_{K_{AB}}(N_B, N_A, B)$ y $MASC = H(AUTH)$.

En este caso hemos sustituido la nueva clave de sesión del protocolo original 2PKDP por un *nonce* aleatorio de B. Teniendo en cuenta que ambos valores tienen las mismas características y solo cambia su finalidad o uso posterior, entonces la seguridad del protocolo modificado es equivalente.

Todos los ataques desarrollados hasta ahora se han basado en repetir respuestas de momentos anteriores, y logar que estas sean aceptadas como nuevas, por lo que estos entran dentro de la categoría de *replay attack*. El mecanismo principal que hemos estado usando en todos los protocolos vistos hasta ahora, para evitar un tipo de ataque tan básico, pero a la vez tan nocivo, es la inclusión de desafíos basados en números aleatorios no repetitivos. Existen otros tipos de parámetros variables con el tiempo que pueden ser utilizados como desafío, y otras ideas para evitar los ataques de tipo *replay attack*. Por ello pasaremos a estudiar estos parámetros y sus posibilidades de uso, así como las otras ideas que evitan este ataque, para luego continuar con el estudio de protocolos más complicados y más seguros.

2.4. Uso de parámetros no repetitivos y variables en el tiempo

Ya hemos visto la importancia de asegurarse de que los mensajes recibidos por cada uno de sus *entes principales*, en una ejecución de un protocolo, sean nuevos y no una repetición de un mensaje de un momento anterior. La medida más común y efectiva para lidiar con esto, es la utilización de parámetros o identificadores variables con el tiempo y no repetitivos (al menos en un periodo prudentemente largo), los cuales son asociados indisolublemente a los mensajes intercambiados cada vez.

Existen distintos tipos de parámetros a considerar para la identificación unívoca de mensajes y la protección contra *replay attack*. Para asegurar que estos briden la seguridad requerida, los parámetros que se usen deben cumplir una serie de condiciones. Estos parámetros variables con el tiempo suelen denominarse *nonce*. Veamos los distintos tipos de *nonce* así como sus características y peculiaridades de uso.

Aunque el término *nonce* es utilizado por lo general para referirse a un *número aleatorio*, existen otros dos tipos de parámetros variables con el tiempo, que pueden ser usados con esta misma finalidad, los cuales son: los *números secuenciales*, y las *marcas de tiempo* (*time-stamps*). En dependencia de la forma en que son utilizados, estos pueden proveer propiedades similares al uso de números aleatorios no repetitivos, aunque cada uno tiene sus particularidades y limitaciones.

Números aleatorios: Como se ha mostrado, los números aleatorios pueden ser usados en los mecanismos de *desafío-respuesta* para garantizar unicidad de los mensajes. La forma en que estos son utilizados, en la autenticación unidireccional, conlleva el intercambio de dos mensajes (Cuadro 2). En el primero de estos, el ente verificador genera un valor aleatorio R usado como *nonce*; luego este valor es retornado con el mensaje después de haber sido asociado al mismo mediante una función criptográfica f con ciertas propiedades requeridas. El ente verificador chequea la respuesta basado en el valor aleatorio que ha generado, y si es correcta deduce que es nuevo, ya que no podría haber sido formada antes de que el *nonce* fuese generado.

Los números aleatorios usados de esta manera permiten establecer una ubicación relativa en el tiempo en la comunicación entre dos partes. Además la propiedad de

aleatoriedad provee incertidumbre, para evitar ciertos tipos de ataques criptográficos sofisticados como lo es el caso del *chosen-text attack* (Colin & Anish, 2003). Señalemos además que cuando hablamos de números aleatorios no descartamos en absoluto a los números pseudo-aleatorios con propiedades de aleatoriedad suficientemente buenas.

Marcas de tiempo: Para garantizar unicidad en los mensajes intercambiados y establecer un momento en el tiempo para estos, el emisor del mensaje añade el valor del tiempo actual al mensaje, a partir de su reloj. El receptor del mensaje compara el tiempo recibido con su tiempo local, si este se encuentra en una ventana de tiempo aceptable con respecto a su tiempo local, entonces el mensaje es considerado como actual. El beneficio de las marcas de tiempo es que se elimina la necesidad de emplear algún mensaje para intercambio de *nonce*, asumiendo que el tiempo local es conocido para ambas partes involucradas en la comunicación, salvo por un error estimado. Esto hace que los protocolos que emplean este tipo de parámetro sean más eficientes en las comunicaciones, debido a que la comprobación de la unicidad de los mensajes requiere de un mensaje menos que en el caso del los mecanismos de desafío-respuesta que usan números aleatorios.

Para ilustrar lo referido mostraremos el siguiente protocolo de autenticación mutua del estándar internacional *ISO/IEC 9798-2* (ISO, 1999), el cual es análogo al protocolo que se muestra en la Cuadro 4, solo que en este caso se usa criptografía de clave simétrica, en vez de una *función hash segura*.

Cuadro 10: Protocolo ISO/IEC 9798-2: autenticación mutua usando marcas de tiempo, tomada de (Colin & Anish, 2003)

1. $A \longrightarrow B:\ \mathcal{E}_{K_{AB}}\{T_A, B\}$
2. $B \longrightarrow A:\ \mathcal{E}_{K_{AB}}\{T_B, A\}$

En este protocolo K_{AB} denota la clave compartida entre los entes A y B. Las marcas de tiempo correspondientes a cada uno se han denotado por T_A y T_B respectivamente. Nótese que con un mensaje menos que en el protocolo de la Cuadro 3 se logra el mismo objetivo.

A pesar de este beneficio observado para las comunicaciones, el uso de marcas de tiempo trae consigo algunos elementos preocupantes para la seguridad; ya que en este caso, la base de la efectividad de este mecanismo, es el establecimiento de una referencia común de tiempo entre las partes que se comunican. Esto requiere que los relojes de todas las partes estén bien sincronizados, y seguros contra modificación de algún atacante. Los desfases entre estos relojes, y las ventanas de tiempo usadas para chequear si una marca de tiempo recibida es correcta, deben ser suficientemente pequeños como para que no abran la oportunidad de un *replay attack*. Además para un uso correcto de este tipo de mecanismos, surge la necesidad de mantener una lista de los valores de tiempo recibidos dentro de una ventana de tiempo, lo cual constituye un inconveniente en relación a la capacidad de almacenamiento y el esfuerzo empleado en la verificación de valores.

Se puede decir que los requerimientos pedidos para la seguridad de los esquemas basados en este tipo de parámetro no son sencillos de mantener. La sincronización perfecta es algo que solo ocurre de forma ideal. Por otra parte, no es tan sencillo determinar cuál margen de error, o desfase entre los relojes, es el más apropiado para una situación específica, y esto no garantiza que los relojes no se salgan de sincronía. Aunque existen diferentes soluciones para sincronizar relojes en sistemas distribuidos, para lograr esta sincronización por medio de protocolos que funcionan sobre redes, estos protocolos en sí, deben ser seguros, y por lo tanto típicamente requieren autenticación; esto lleva a una interdependencia que más bien parece un ciclo repetitivo.

Más aún, se ha planteado que si el reloj de un ente principal en un protocolo basado en marcas de tiempo, avanza más allá del tiempo de sincronía del resto del sistema, una vulnerabilidad inmediata se abre, incluso cuando el tiempo de este reloj haya sido corregido. Esto se debe a que un atacante puede haber capturado y suprimido un mensaje que se volverá actual en el futuro. Este ataque ha sido llamado, *suppress-relay attack* (Colin & Anish, 2003).

Números secuenciales: Sirven como identificadores únicos para los mensajes. En este caso, ambas partes deben mantener una secuencia sincronizada cuyo valor es enviado junto al mensaje, criptográficamente ligado, y si la verificación en base al valor actual es exitosa, entonces ambas partes generan el próximo valor de la secuencia, manteniendo así

la sincronización. Por lo general se escogen secuencias monótonas para garantizar la no repetitividad de valores, lo cual sigue siendo en este caso un requisito indispensable. Bajo esta política, si la respuesta enviada con el valor actual del emisor coincide con la calculada por destinatario, en base al valor que este mantiene, el mensaje será considerado como nuevo, ya que el valor actual de la secuencia no podría haber sido de uso anterior. De este modo si la secuencia se mantiene en secreto, puede servir como base para la autenticación de entidades, aunque posee el inconveniente de que la predictibilidad de la mayoría de estas secuencias podría poner en riesgo la seguridad del protocolo. Una secuencia solo debe ser usada en asociación a un solo par de entidades, y es recomendable que se usen secuencias distintas para cada una de las direcciones en que los mensajes son enviados, o sea, de ser posible, una secuencia para cada remitente.

Si ambas partes de una comunicación pierden la sincronización sobre la secuencia mantenida, entonces es necesario aplicar un procedimiento para acordar un nuevo valor inicial común. Este tipo de métodos para recuperar la sincronía se deben aplicar con cuidado para no caer de nuevo en valores ya generados anteriormente, con lo cual se repetirían valores de la secuencia. No es difícil verificar, que si una secuencia no está sincronizada entre las partes que deben mantener su estado, entonces es posible realizar un *replay attack* contra aquella parte que se ha quedado en un estado anterior (Colin & Anish, 2003).

Una última aclaración es importante sobre este tipo de *nonce*, y es que, a diferencia de la sincronización por estampas de tiempo, en este caso, el paso de un valor a otro de la secuencia está determinado por la ocurrencia de algún tipo de evento.

Después de haber visto las características más importantes de los tres tipos de *nonce* que se usan en los mecanismos de autenticación, hagamos algunas comparaciones basadas en el uso práctico de los mismos.

Observemos que la razón por la que los mecanismos de *desafío-respuesta* basados en números aleatorios no pueden ser completados en un solo paso, es que se hace necesario que el parámetro aleatorio sea controlado por el mismo verificador, ya que de lo contrario este se arriesgaría a recibir una respuesta asociada con un valor que ha sido usado antes, pues estos parámetros aleatorios son olvidados después de ser completado el protocolo.

De esta forma se tiene que la respuesta no debe ser enviada junto al desafío correspondiente, como por ejemplo se ilustra en el protocolo erróneo, Cuadro 11.

Cuadro 11: Forma errónea de usar un esquema de desafío-respuesta basado en números aleatorios

1. $A \longrightarrow B : A , R_A , MAC_K(A , R_A)$
2. $B \longrightarrow A : B , R_B , MAC_K(B , R_B)$

Sin embargo, esto no es así en el caso los otros dos parámetros variables analizados, pues hemos visto cómo en el caso de las marcas de tiempo, cada ente incluye su estampa de tiempo en el mensaje usado para la autenticación, y así cada uno usa un solo mensaje. Por ello si ahora reemplazamos el desafío aleatorio por una marca de tiempo en el protocolo de la Cuadro 11, este ya no presentará el problema señalado inicialmente.

Cuadro 12: Protocolo basado en marcas de tiempo usando una función MAC.

1. $A \longrightarrow B : A , T_A , MAC_K(A , T_A)$
2. $B \longrightarrow A : B , T_B , MAC_K(B , T_B)$

Aquí la autenticación es llevada a cabo usando los valores en claro; y el código $MAC_K(\cdot)$ es necesario para proteger la integridad de estos valores. A diferencia del caso de *desafío-respuesta* usando números aleatorios, el uso del *nonce*, en este caso, es inseguro sin un mecanismo de autenticación de mensaje, como lo es en este caso el valor MAC agregado en el mismo mensaje.

Supongamos ahora que se quiere usar una secuencia numérica S_n sincronizada para la autenticación, en vez de números aleatorios o marcas de tiempo. Esta vez A y B deben mantener una sincronización exacta sobre la secuencia, para poder autenticarse exitosamente, y bajo este supuesto no es necesario, sino más bien imprudente, enviar en claro el valor actual de la secuencia que mantienen ambas partes. Así, el valor del *nonce* es usado implícitamente, a diferencia de los otros dos casos. El protocolo anterior quedaría de la siguiente manera con números secuenciales.

Cuadro 13: Protocolo basado en números secuenciales usando una función MAC.

1. $A \longrightarrow B : A, MAC_K(A, S_n)$
2. $B \longrightarrow A : B, MAC_K(B, S_{n+1})$

El valor $MAC_K(\cdot)$ es siempre usado para la autenticación, como prueba del conocimiento del secreto compartido K, el cual se mantiene fijo entre ambas partes, y por otra parte las buenas propiedades criptográficas de este tipo de funciones garantizan la protección de este secreto y la seguridad de estos esquemas contra ciertos ataques de criptoanálisis. Llamemos la atención sobre el hecho de que no es apropiado para la seguridad del protocolo que se repita el uso de algún valor secuencial; de este modo, en el caso del ejemplo del Cuadro 13, en la próxima ejecución del protocolo el ente A deberá usar el valor S_{n+2} y no el valor S_{n+1} ya utilizado por su contraparte.

Por último aclaremos que, aunque los protocolos de los Cuadros 11, 12 y 13 tienen fallas de seguridad, nuestra intención ha sido usarlos para ilustrar, lo mejor posible, las particularidades en la forma de uso de cada tipo de *nonce*.

2.5 La idea mixta del Valor de Estado Semi-Aleatorio

Si realizamos una comparación entre los distintos valores de *nonce* expuestos; los números aleatorios, por ser impredecibles, suelen proporcionar un mayor nivel de seguridad que los demás tipos de valores, a cambio de un mayor intercambio de mensajes. Por otra parte, los mecanismos basados en *marcas de tiempo* y *números secuenciales* son más eficientes en cuanto a las comunicaciones; sin embargo, requieren de sincronización, y mecanismos de recuperación de la sincronía en caso de que esta se haya perdido, lo cual ocurre con alta probabilidad. Estos mecanismos de sincronización suelen ser delicados sobre todo en el caso de las *marcas de tiempo*. Por ese motivo, este tipo de *nonce* es evitado por muchos de los diseñadores de protocolos actuales. De otra forma, los mecanismos de sincronización de las *secuencias numéricas* no son generalmente tan problemáticos; sin embargo, generalmente estas pueden ser

determinadas una vez que son conocidos unos pocos valores, por lo que las mismas deben ser usadas con cuidado para no comprometer la seguridad. De cualquier forma, se ha señalado cómo la pérdida de sincronía, en los mecanismos basados en estos dos últimos tipos de *nonce*, abre la posibilidad de un *replay attack*. Esto se debe a que, en una pérdida de sincronía, algunos valores usados podrían ser aceptados como nuevos por alguna de las partes en un momento posterior.

En este epígrafe exponemos una idea para la generación de otro tipo de *nonce*, que no ha sido encontrada en ninguna de las bibliografías revisadas hasta el momento para la conformación de esta tesis y que será luego utilizada en nuestra propuesta para abaratar el costo computacional en los cambios que efectuaremos al protocolo de Vaidya et al. (2011). Mediante este nuevo *nonce* pretendemos combinar la ventaja de mantener una secuencia sincronizada entre dos entes principales, con la incertidumbre proporcionada por los números aleatorios.

La idea consiste en formar una secuencia, a partir de un valor secreto inicial y una secuencia totalmente aleatoria, construida sobre la marcha de las comunicaciones, haciendo depender cada valor actual de la secuencia propuesta, del valor anterior de esta y un nuevo valor aleatorio generado recientemente. O sea, que en vez de seguir el esquema de *desafío-respuesta* estándar y desechar los valores aleatorios que van siendo generados, estos serán incorporados progresivamente a nuestra secuencia, proporcionando así sincronización secuencial y aleatoriedad. A los valores de esta secuencia hemos decidido llamarlos *valores de estado semi-aleatorios*, aludiendo al hecho de que están formados a partir de un valor acordado y una secuencia de números aleatorios, y teniendo en cuanta que debe mantenerse de forma sincronizada.

El planteamiento formal de esta idea sería como sigue.

Definición 2.4: Sea $\{R_n\}_{n\geq 0}$ una sucesión números aleatorios y S un valor semilla (*seed*) que constituye un secreto compartido, entonces definimos la secuencia de valores de estado $\{E_n\}_{n\geq 0}$ mediante la siguiente fórmula recurrente:

$$E_0 = H(S)\ ,\ E_{n+1} = \Phi(E_n\ , R_n) \qquad (1)$$

Donde H representa una *función hash segura* y $\Phi(\cdot)$ puede ser una combinación de operaciones criptográficas ligeras que actúan sobre el par de valores E_n y R_n, o una

operación criptográfica simple entre estos dos valores, y debe ser una función fuertemente resistente a colisiones. El valor inicial E_0 bien puede definirse de algún otro modo conveniente siempre y cuando se preserve la confidencialidad del secreto compartido S.

Ilustremos mediante un ejemplo una de las posibles formas de uso que podría tener este tipo de secuencia en la autenticación mutua. Supongamos que los entes A y B han acordado mantener la secuencia de valores $\{E_n\}_{n\geq 0}$ (tal y como la hemos definido) de forma sincronizada, y que esta es guardada en secreto por ambas partes. Asumamos además que ambos se encuentran actualmente en el estado E_0. Un esquema similar al de los protocolos correspondientes a los Cuadros 12 y 13, puede lograrse, usando este nuevo *nonce* de la siguiente forma:

Cuadro 14: Protocolo ejemplo del uso del valor de estado semi-aleatorio para la autenticación mutua

1. $A \longrightarrow B : A\ ,\ R_0\ ,\ H(E_0, A)$
2. $B \longrightarrow A : B\ ,\ R_1\ ,\ H(E_1, B)$

Inicialmente, el identificador de A será ligado criptográficamente con el valor de estado actual E_0, y se envía el valor aleatorio R_0 el cual servirá para el computar el próximo valor de estado E_1 por ambas partes. El estado inicial puede ser visto como un secreto compartido entre A y B, ya que este ha sido computado de igual forma a partir de la semilla S la cual es un secreto compartido entre los entes A y B. De este modo, A usa el secreto E_0 para autenticarse con B y usa el valor aleatorio generado como un desafío. El ente B calcula el valor $H(E_0, A)$ usando su valor de estado almacenado, y comprueba la validez del mismo. Si el valor recibido es correcto B autentica al ente A, bajo el supuesto de que solo A pudo haber encriptado este identificador correctamente usando como clave el estado compartido. Luego usa el valor R_0 para computar el próximo valor de estado $E_1 = \Phi(E_0\ , R_0)$. Este próximo valor de la secuencia será entonces utilizado por B para autenticarse con A usando un mensaje del mismo tipo, o sea $H(E_1, A)$. Dado que A generó el valor R_0 este puede calcular la respuesta esperada por su parte y compararla

con la recibida para verificar si la misma es correcta, con lo cual se autenticaría positivamente a B. El valor aleatorio R_1 es incorporado a la respuesta de B como un desafío para la próxima ejecución del protocolo, que se realizará en base al estado $E_2 = \Phi(E_1, R_1)$; y así sucesivamente.

Es impotente hacer la observación de que, aún si la secuencia de valores aleatorios $R_1, R_2, \dots, R_n$ fuese observada por un adversario, este no podrá obtener nunca el próximo valor E_{n+1}, siempre que los valores de estado anteriores se hayan mantenido en secreto.

Este esquema análogo a los presentados en los Cuadros 12 y 13 resulta de igual forma incompleto, presentando igualmente una serie de fallas de seguridad. Sin embargo, a este nivel se logran algunas ventajas con su aplicación. Por ejemplo, notemos que aunque de cierta forma hemos utilizado desafíos basados en números aleatorios en un esquema similar a la forma errónea planteada en el Cuadro 11., esta vez sí funciona el procedimiento debido a la sincronización. De este modo se puede realizar la autenticación mutua con un mensaje menos que en el caso de del mecanismo de *desafío-respuesta* mediante números aleatorios. Otra ventaja está relacionada con la recuperación de la sincronía. Lo más usual es que, si en un mecanismo de *desafío-respuesta* basado en sincronización secuencial se pierde la sincronía, de modo que el ente A mantiene el valor S_n y el ente B mantiene el estado S_m, entonces uno de los entes debe intentar autenticarse con el otro mediante los estados intermedios, recorriendo los mismos hasta que lleguen a un estado común. En el caso de la autenticación usando el *valor de estado semi-aleatorio*, solo sería necesario generar otro valor inicial E_0 a partir del secreto compartido y un número aleatorio actual. A partir de entonces, con la ayuda de la sucesión aleatoria y escogiendo la función $\Phi(\cdot)$ de forma adecuada, se garantiza que no se repetirá una secuencia de valores generados anteriormente para la autenticación. Esto resulta más eficiente que en otros casos que requieren recuperación de sincronía. Por otra parte este tipo de mecanismo contiene mayor nivel de incertidumbre que las secuencias deterministas, y se aporta mayor protección al secreto compartido, sobre todo si este es un secreto débil como es el caso de una contraseña.

Para incrementar el nivel de seguridad, la secuencia aleatoria, además de funcionar como factor de cambio del valor de estado, puede usarse con fines de enmascaramiento.

Agregando además el valor *hash* de los valores aleatorios para poder verificar su integridad, obtenemos una versión mejorada del esquema expuesto en la Cuadro 14.

Cuadro 15: Protocolo mejorado usando el valor de estado semi-aleatorio para la autenticación mutua.

1. $A \longrightarrow B : A\ ,\ R_0 \oplus H(E_0, A)\ , H(R_0)$
2. $B \longrightarrow A : B\ ,\ R_1 \oplus H(E_1, B)\ , H(R_1)$

La verdadera aplicación concreta de este nuevo *nonce* que introducimos en este trabajo, se verá a través de su uso posterior dentro de las modificaciones que propondremos al protocolo de Vaidya et al. (2011), con lo cual se logrará un nivel de seguridad aceptable y un aumento en la eficiencia del mismo.

2.6 Protocolos de autenticación basados en contraseñas

En los casos anteriores hemos estudiado protocolos de autenticación mutua, basados en claves secretas compartidas, en los cuales intervienen solo dos entes que asumen el mismo tipo de *rol*. También hemos visto lo peligrosa que puede ser la simetría en un esquema, para la autenticación segura.

Realmente nos interesa estudiar esquemas en los que los principales asumen diferentes tipos de *roles*. Unos de los esquemas de autenticación a los que más atención se les ha prestado, dado su uso creciente, es aquellos en que se necesita acceder de forma remota a un Sistema Informático, a partir de un Terminal, operado por un sujeto de confianza. Como es el caso específico de un *Usuario* que requiere autenticarse con un *Sistema* (o *Servidor*). En este caso, el *Usuario* es quien requiere ser autenticado, para acceder a servicios o ciertos datos protegidos. Solo uno de estos dos principales será el iniciador del protocolo en todas las ocasiones en que este puesto en práctica. Este tipo de esquema asimétrico, proporciona ciertas ventajas en el diseño de protocolos seguros, en comparación con lo estudiado previamente.

Para empezar señalemos que la asignación de *roles* distintos hace imposible un *interleaving attack*, tal como los que hemos visto hasta ahora, ya que el adversario tendría que iniciar sesiones paralelas con ambos principales, simulando ser su ente par. Por las mismas razones también evita de por si un *refelction attack.* Por ello desecharemos la posibilidad de este tipo de ataques dentro del los próximos protocolos basados en este modelo *Usuario-Sistema*.

La autenticación mediante una clave compartida se basa en la posesión de una clave aleatoriamente generada en un domino que puede sobrepasar los 100 bits. Sin embargo los usuarios comunes tienen problemas para recordar claves secretas de este tipo, con suficiente entropía. Ya hemos señalado como la forma preferida de autenticación en los escenarios orientados a servicios de usuarios, es aquella basada en contraseñas. Sin embargo, a simple vista puede parecer imposible el hecho de poder brindar seguridad basado en un secreto débil como lo son las contraseñas, las cuales son susceptibles a ataques de fuerza bruta. Por ello, no es de extrañarse que los primeros protocolos basados en contraseñas no hicieran su aparición en la literatura hasta la década de los 1980s (Colin & Anish, 2003). La mayoría de estos primeros protocolos usaban criptografía de clave pública. En los mismos la contraseña del usuario era usada como clave, o como base para generar una clave.

En los protocolos con los que trataremos a continuación, enmarcados en el escenario *Usuario-Sistema*, las acciones del *Usuario*, durante la ejecución de un protocolo son distintas de las acciones del *Sistema*. Específicamente el *Usuario* siempre será poseedor de una contraseña en claro *PW* (abreviatura del término anglosajón *PassWord*) mientras que el *Sistema* podría almacenar sólo una imagen de la misma mediante una función unidireccional. Usaremos la letra *U* para denotar a un *Usuario* de un sistema que actúa como solicitante, requiriendo ser autenticado; y la letra *S* para denotar al *Sistema* (o *Servidor*) que actuará como verificador.

Por el momento asumiremos que las contraseñas que se usan en los protocolos han sido debidamente escogidas, de modo que son fuertes, con una alta entropía, una suficiente cantidad de caracteres y una muy baja probabilidad de aparecer en un diccionario. Esto, aunque no impide del todo un ataque de fuerza bruta, lo dificulta

considerablemente, y disminuye la probabilidad de éxito de los ataques basados en diccionario. A pesar de ello no descartaremos del todo ninguno de estos dos ataques.

2.7 Contraseñas de un solo uso (One Time Passwords)

Ya se analizaron previamente las numerosas vulnerabilidades e inconvenientes que traen consigo el uso de las contraseñas estáticas, lo cual nos conduce a un nuevo modelo de uso de las contraseñas, en el que se introduce dinamismo a las mismas. De este modo, lo más sensato es que las contraseñas deben cambiar con el paso del tiempo. Dentro de este tipo de paradigma la solución más segura es la construcción de una sucesión de contraseñas, en la cual se use una distinta para cada ejecución del protocolo de autenticación. Una forma de lograr este dinamismo es mediante la introducción de parámetros que controlan el paso de un valor a otro. Para ello siempre es necesario algún tipo de coordinación o sincronía entre los entes que participan en la comunicación. En este tipo de esquemas se suele usar la contraseña como parte de la semilla para el algoritmo de generación de valores, la cual es preferentemente de unos 20 bytes. Es importante que los valores generados oculten lo mejor posible la semilla, ya que esta constituye un secreto, que funciona como base para la seguridad del mecanismo de autenticación. De igual forma es necesario que no se puedan obtener los próximos valores de la secuencia a partir de un valor dado, sin el conocimiento del valor secreto compartido entre las partes genuinas. De esta forma, si una contraseña se ve comprometida esta no podrá ser utilizada por el adversario para próximas autenticaciones. Bajo las condiciones planteadas diremos que la sucesión de valores generada para la autenticación es segura.

Definición 2.5: A una sucesión de contraseñas de un solo uso, una para cada ejecución del protocolo de autenticación, se les denomina *One Time Passwords* (contraseñas de solo uso), y nos referiremos a ellas a partir de ahora de forma abreviada como OTPs.

Con el objetivo de mejorar la seguridad de los mecanismos de autenticación basados en contraseñas, las sucesiones de OTPs combinan desafíos implícitos o explícitos,

generalmente basados en información pública, con secretos compartidos, como son las contraseñas de usuario. Esta combinación se realiza mediante operaciones criptográficas. Así se logra inyectar factores de incertidumbre a estos mecanismos y al mismo tiempo se obtiene una contraseña única cada vez.

Muchos de los mecanismos de autenticación mediante OTP se han propuesto para la autenticación segura a través entornos de red abiertos. En muchos de estos, los costos computacionales en el lado del *Usuario* son altos, implican altos costos de comunicación, o asumen una cantidad limitada de autenticaciones. Debido a que los sistemas basados en clave pública suelen ser computacionalmente costosos, nos centraremos en este caso en sistemas de clave simétrica, que utilizan operaciones ligeras como son las funciones *Hash* y operaciones *XOR*. Esto los hace más apropiados para una variedad más amplia de escenarios, incluyendo las redes inalámbricas.

A continuación exponemos las variantes fundamentales de OTPs para la autenticación segura en el escenario *Usuario-Sistema*:

1. *Basado en una lista pre-compartida:* El *Usuario* y el *Sistema* comparten una lista de valores independientes entre sí. El *Usuario* elige un valor distinto de esta secuencia para autenticarse cada vez, y el *Sistema* verifica que el valor recibido se encuentra entre los valores no usados. Luego de esto el valor usado es desechado y se repite el proceso para un nuevo valor. Este método se usó en los sistemas iniciales de OTPs, y los usuarios se veían forzados a conservar una extensa lista de valores, lo cual es muy poco práctico y ha quedado obsoleto.
2. *Basado en sincronización de tiempo*: Este usa el reloj del dispositivo de *Usuario* sincronizado con la el *Sistema* para generar una sucesión de contraseñas. Cada contraseña cambia con una frecuencia determinada, a intervalos iguales de tiempo. Esta sucesión está basada en una semilla secreta compartida y el valor de tiempo actual; a los cuales se les aplican operaciones criptográficas cuyas salidas tengan aspecto aleatorio para mantener así la seguridad de este tipo de esquemas. Estos valores tienen un corto tiempo útil definido por el periodo de tiempo que transcurre entre dos valores consecutivos del algoritmo generador. De este modo, si un valor actual es comprometido, el margen de tiempo en el que este puede ser

usado por un atacante es muy limitado. Por lo general el periodo de tiempo establecido entre un valor y su consecutivo es de 2 minutos.

Un inconveniente de este método es que una contraseña puede ser enviada en un momento en que el valor de la otra parte ha expirado, y por tanto en este instante no coincidir con el mismo. De igual forma es necesario que los relojes de ambas partes estén muy bien sincronizados, lo cual no siempre se puede lograr de forma permanente. Un ejemplo representativo de este tipo de mecanismo es el TOTP, el cual puede ser encontrado en el RFC 6238.

3. *Basado en sincronización secuencial*: Este modo de generación suele estar asociado a la ocurrencia de algún evento determinado, como puede ser el presionar un botón de un dispositivo; por lo que también se le conoce en la literatura como generación basada en evento (*event-base OTPs*). En este caso se mantiene un contador sincronizado entre el *Usuario* y el *Sistema*, cuyos valores se utilizan para generar la secuencia de contraseñas, partiendo de un secreto compartido y por medio de operaciones criptográficas. Cada vez que una autenticación es completada, cada una de las partes se encargará de que su contador pase al próximo valor. La parte iniciadora del protocolo generará la nueva contraseña a partir del valor actual del contador una vez que el evento generador sea ejecutado; mientras que la parte verificadora podrá computar por su parte el valor correcto correspondiente al valor actual de su contador, y compararlo con el valor recibido. Si estos coinciden la autenticación será positiva. Una autenticación fallida puede deberse a falta de sincronía, por lo que en este caso se debe identificar si es esta la cusa, e implementar un mecanismo de recuperación de la sincronía. Para ello, la parte más rezagada en la generación de las contraseñas, debe generar una cantidad de valores subsiguientes a su valor actual, hasta lograr una coincidencia con su contraparte; luego de esto, ambos entes posicionarán su contador en el valor correspondiente a la contraseña acertada. Un ejemplo importante de este tipo de mecanismos lo constituye el HOTP (M’Raihi, Bellare, Hoornaert, Naccache, & Ranen, 2005), el cual expondremos luego.

4. *Basado en desfío-respuesta*: Este método funciona parecido al esquema tradicional de desafío respuesta usando números aleatorios, que hemos visto hasta ahora. El ente verificador, que suele ser el *Sistema*, envía al *Usuario* algún dato que funciona como desafío, y este lo combina con el secreto compartido mediante operaciones criptográficas, para componer así la respuesta esperada, la cual funciona como una contraseña nueva para el proceso de autenticación actual. Esta respuesta es envida de vuelta al *Sistema*, el cual computará por su parte la respuesta correcta para compararla con la recibida por parte del *Usuario*. Si estos dos valores coinciden el *Usuario* es autenticado positivamente por el *Sistema*. La autenticación en el sentido inverso es realizada de forma similar. En este caso el intercambio de información es mayor, sin embargo no es necesario preocuparse por las implicaciones que traen consigo los mecanismos que requieren sincronización.
5. *Basado Cadenas de Hash*: Una *cadena de hash* puede ser obtenida mediante la aplicación sucesiva de una *función hash segura* a un valor fijo s, el cual es conocido como semilla. En este tipo de esquemas, la sucesión de contraseñas está formada por los valores de la cadena recorridos en orden descendiente de la cantidad de operaciones *hash* computadas, tal y como se ilustra en la expresión 2.

$$\underset{\longrightarrow}{H^N(s)} \prec H^{N-1}(s) \prec \underset{\longrightarrow}{H^{N-2}(s)} \prec \cdots \prec \underset{\longrightarrow}{H(s)} \prec s \qquad (2)$$

Donde hemos utilizado $\prec$ como una relación de orden, y $H^N(s)$ representa la aplicación sucesiva de la función H sobre el valor s, tal como se indica a continuación.

$$H^N(s) = H\left(H\left(H(\cdots H(s))\right)\right), \qquad N - veces$$

El uso de las *cadenas de hash* para la autenticación fue propuesto por primera vez por Lamport, para la autenticación de usuarios como una solución que evita la efectividad de ataques de *eavesdropping*, y busca asegurar el secreto compartido, en caso de robo del verificador almacenado por el *Sistema* (Lamport, 1981). Este

tipo de métodos es bastante eficiente en las comunicaciones, y puede ser visto como un mecanismo de desafío respuesta con sincronización, en el cual el desafío está definido por la posición actual de la cadena. La seguridad que brinda el uso de las *cadenas de hash* se basa en el hecho de que, conocido el valor $H^N(s)$ no puede ser generado el valor $H^{N-1}(s)$ por aquellos que no conocen el valor s, debido a la propiedad unidireccional de la *función hash* utilizada. Por otra parte, en su uso original, el *Sistema* solo guarda un *hash* de la contraseña válida en cada momento, y el proceso de renovación sucesiva de las contraseñas ocurre de forma tal que cada contraseña de usuario es el valor que necesita el *Sistema* para verificar la próxima contraseña; de este modo se evita que queden comprometidos los próximos valores en caso de robo del verificador en el *Sistema*. Es importante señalar que la aplicación sucesiva de una operación *hash* segura a un valor fijo, produce una secuencia de valores distintos entre sí; de lo contrario se contradecirían la propiedades 1), 2) y 3) de las *funciones hash* enunciadas en la página 14 del Capítulo 1.

Las características mencionadas han hecho que este tipo de mecanismo haya cobrado auge, siendo cada vez más usado en variedad de protocolos de autenticación robusta, y otros tipos de protocolos de seguridad. Sin embargo las *cadenas de hash* poseen dos inconvenientes importantes:

i. Limitan la cantidad de autenticaciones que pueden ser realizadas con una semilla secreta compartida s, ya que se fija de inicio el valor más alto de la secuencia, a partir del número N. Esto implica que luego de recorrida la cadena completa, es necesario volver a establecer un secreto nuevo que sirva para generar una nueva secuencia.

ii. Si se escoge un valor de N muy alto, para retrasar la ocurrencia del evento anterior, entonces una de las partes (y en algunos casos ambas) debe incurrir en un costo computacional elevado, en comparación con los mecanismos que usan criptografía ligera. Esto se debe a que los valores generados en la cadena van en orden decreciente de la cantidad de operaciones *hash* sucesivas, comenzando por el mayor valor H^N.

2.8 HOTP: algoritmo de one-time password basado en HMAC

HOTP es un algoritmo de bajo costo para la generación de OTPs, basado en HMAC, que ha sido propuesto por la OATH (*initiative for Open AuTHentication*). El mismo se encuentra descrito en el RCF 4226 del IETF (M'Raihi et al., 2005). Sus características convenientes han provocado que este sea últimamente bastante usado en los protocolos actuales de autenticación robusta; y en particular es usado dentro del protocolo de Vaidya et al. (2011) que será objetivo de nuestro análisis.

Se basa en un contador ascendente y una clave simétrica para la generación de los valores OTP, por lo que funciona como un mecanismo de sincronización secuencial para la autenticación robusta. El contador es usado en sustitución del mensaje correspondiente en el algoritmo HMAC, mientras que la clave simétrica es conocida solo para el cliente y el servidor implicados. Normalmente la *función hash segura* SHA-1 es usada para obtener los valores del algoritmo HMAC, pero esta podría ser reemplazada por cualquier otra *función hash* con requerimientos de seguridad mejores, en caso de que exista una disponible y sea deseable.

El algoritmo opera de la siguiente forma. Inicialmente la función generadora de SHA-1 es inicializada a partir del secreto compartido. A continuación se calcula el *valor hash* del contador y se le aplica un truncamiento dinámico para extraer solo ciertos bytes de esta cadena de salida del HMAC. Finalmente de esta cadena resultante se extrae el resto módulo 10^n, donde n es el número de dígitos deseados en el resultado final, para ser usados como OTP. El proceso de cálculo indicado puede ser expresado por medio de la fórmula:

$$HOTP(K, C) = Trunc\big(HMAC\_SHA1(K, C)\big)_n \quad (3)$$

donde $Trunc$ representa la función de truncamiento que convierte la salida de $HMAC\_SHA1$ en un valor de OTP; mientras K y C representan la clave secreta compartida y el contador ascendente respectivamente. Para el correcto funcionamiento del algoritmo, tanto el cliente como el servidor involucrados en este deben generar el mismo valor de OTP, partiendo del hecho de que ambos mantengan el mismo valor del contador de forma sincronizada.

Una vez vistos y analizados los mecanismos de autenticación robusta más importantes para este trabajo estamos preparados para el análisis de ciertos protocolos complejos, los cuales serán presentados en el próximo capítulo de forma progresiva para dar lugar al protocolo en el que realmente nos interesa abundar.

CAPÍTULO 3: EL PROTOCOLO DE VAIDYA ET AL. (2011), RESULTADO DE LA EVOLUCIÓN DE LA AUTENTICACIÓN ROBUSTA.

En el presente capítulo expondremos una secuencia de protocolos de autenticación robusta de forma que se evidencie progresivamente la evolución del los mismos hacia esquemas cada vez más seguros. A lo largo de este recorrido se irán incorporando técnicas, procedimientos y características, que irán incrementando paulatinamente la complejidad de los protocolos que se exponen. Como resultado de esta evolución, que busca alcanzar niveles de seguridad cada vez mejores para cumplir con los retos de cada momento, se presenta finalmente el protocolo de Vaidya et al. (2011).

3.1 Evolución de los esquemas de autenticación robusta mediante OTPs

(a) Esquemas básicos usando cadenas de hash

Dos de las formas más comunes en que un atacante puede obtener la contraseña de un usuario de un sistema son:

1. Mediante el acceso fraudulento a la información que el *Sistema* almacena sobre los usuarios.
2. Espiando las comunicaciones del *Usuario* en modo *eavesdropping*.

En 1981, Lamport publica un trabajo donde se propone resolver simultáneamente estos dos tipos de vulnerabilidades mediante la introducción de un esquema, novedoso para su tiempo, que utiliza una contraseña distinta para cada autenticación, dando lugar así las contraseñas de un solo uso (OTPs) que ya hemos descrito (Lamport, 1981).

La idea planteada por Lamport es la siguiente: El *Usuario* debe usar una secuencia de valores distintos $x_1, x_2, x_3, \dots, x_{N-1}, x_N$, donde N es un número suficientemente alto como para que el *Usuario* no agote todas las contraseñas de la sucesión en un tiempo

prudencialmente largo. Mientras tanto, el *Sistema* almacenará la secuencia de valores $y_1, y_2, y_3, \dots, y_{N-1}, y_N$, correspondiente a los *valores hash* de cada posición de la sucesión almacenada por el *Usuario*; de forma que se cumple la relación $y_i = H(x_i)$, $\forall\, i = \overline{1,N}$. En este caso el *Usuario* le enviará al *Sistema* el valor x_i en la $i-esíma$ autenticación, y el sistema comprobará la validez del valor recibido calculando un *hash* del mismo y comparando este resultado con el valor y_i almacenado ($H(x_i) \overset{?}{=} y_i$). De este modo, si un atacante logra leer el archivo del sistema con la información de verificación no obtendrá directamente las contraseñas de los usuarios, y por otra parte si el atacante logra interceptar la contraseña x_i, esta no le servirá para la próxima autenticación. Sin embargo, este método obliga al *Usuario* y al *Sistema* a guardar y mantener una lista muy extensa de valores por largo período.

Lamport plantea entonces la mejor idea para resolver el problema de forma óptima, la cual consiste en que ambas sucesiones mantenidas sean una *cadena de hash*. O sea, que partiendo de un secreto s, que funciona como semilla, la sucesión de contraseñas que corresponde al *Usuario* sea tal que $x_i = H^{N-i}(s)$ y $x_i = H(x_{i+1})$, lo cual, en conjunto con las relaciones anteriormente planteadas, da lugar al siguiente esquema:

$$\begin{array}{ccccccc} & x_1 & & x_2 & \cdots & x_{N-1} & x_N \\ H^N(s) & H^{N-1}(s) & H^{N-2}(s) & \cdots & H(s) & s & , \\ y_1 & & y_2 & & y_3 & \cdots & y_N \end{array} \qquad (4)$$

donde las sucesión mantenida por el *Sistema* está desfasada con la sucesión de contraseñas del *Usuario* en una posición ($x_i = y_{i+1}$), que corresponde a la aplicación de la *función hash* solo una vez. De esta forma, para la $i-esíma$ autenticación, el *Sistema* solo tiene almacenado el valor y_i, y el *Usuario* solo debe almacenar o recordar el secreto s. Una vez que este quiere ser autenticado por el *Sistema*, calcula el valor x_i, que enviará para su autenticación. El *Sistema*, al recibo de la nueva contraseña, procede a verificar su validez, calculando un *hash* del la misma y lo comparándolo con el valor almacenado y_i; si estos dos coinciden, el *Sistema* reemplaza el valor y_i mantenido por el valor $x_i = y_{i+1}$. Así ninguna de las partes necesita almacenar una larga lista de valores sino solo mantener una sincronización respecto a la posición de la *cadena de hash*. En adición a lo planteado,

nótese que la contraseña en claro, o una función de la misma, podrían ser usadas como valor inicial *s* para la generación de la *cadena de hash*.

Las características, ventajas e inconvenientes de estas *cadenas de hash* han sido ya analizadas anteriormente en la sección 2.7. En el momento en que este tipo de esquema fue introducido no parecían estar creadas las condiciones tecnológicas para su correcta implementación, pues el *Usuario* no podría generar la sucesión de contraseñas por sí mismo, sino que necesita apoyarse en algún dispositivo capaz de realizar operaciones criptográficas sencillas. Hubo que esperar a principios de los años 1990's para que apareciera el primer esquema de autenticación con uso práctico, basado en la idea de Lamport de las *cadenas de hash*.

El mecanismo de autenticación robusta mediante contraseñas de un solo uso llamado S/Key, fue propuesto por Neil M. Haller como resultado de sus investigaciones desarrolladas en los laboratorios de *Bell Communications* (Haller, 1995). Este fue diseñado para sistemas Unix basándose completamente en la idea de las *cadenas de hash* anteriormente planteada por Lamport; no obstante, el diseño de dicho mecanismo es independiente de cuestiones específicas de este tipo de sistemas, pudiendo ser fácilmente adaptable a toda una variedad de otros entornos. Así, es posible realizar la implementación del mismo en software, o en algún dispositivo de propósito particular en la parte del *Usuario*, como puede ser un terminal de bajo costo, una tarjeta inteligente o incluso alguna calculadora capaz de hacer operaciones criptográficas.

El objetivo fundamental de S/Key es proteger al *Usuario* contra ataques pasivos, como es el caso del *eavesdropping attack*, en el sentido de que si un atacante es capaz de monitorear las comunicaciones, la información que este pueda recopilar no le sea inútil a los efectos de intentar ganar acceso al *Sistema*. La seguridad de dicho mecanismo está enfocada en la fase de *inicio de sesión*. El mismo funciona a partir de una frase secreta (*pass phrase*) de cualquier longitud mayor que 8 caracteres, escogida por el *Usuario*, la cual es usada para generar la *cadena de hash*. Esta *frase secreta* nunca atraviesa la red, y cada contraseña generada por el algoritmo está asociada a solo un inicio de sesión; de modo que S/Key es seguro contra *replay attacks*.

Una mejora que se introduce en S/Key a la idea inicial propuesta por Lamport consiste en que, antes de generar la *cadena de hash*, la clave secreta es combinada con una semilla $Seed$ aleatoria enviada en claro por el *Servidor* para producir el valor generador s de esta cadena. Esto permite reciclar dicho valor, de forma tal que el *Usuario* pueda usar el mecanismo de autenticación desde distintos terminales o dispositivos con su misma *frase secreta* actual.

A continuación ilustramos el mecanismo S/Key a través de los mensajes intercambiados y siguiendo la misma notación compacta que hasta el momento.

Cuadro 16: Esquema del mecanismo de autenticación S/Key basado en OTP, mediante cadenas de hash

$R.\ Sistema \longrightarrow Usuario : Seed\ ,\ N$	(*fase de Registro con el Sistema - R*)
$L\#1.\ Usuario \longrightarrow Sistema : H^N(s)$	(*fase de Inicio de Sesión – Login - L*)
$\vdots \qquad \vdots \qquad \vdots$	
$L\#i\ .\ Usuario \longrightarrow Sistema : H^{N-i}(s)$	($i-$ ésimo *Inicio de Sesión*)

El valor generador s es obtenido del siguiente modo: se calcula una ronda de *hash* MD4 a la *frase secreta* concatenada con la semilla $Seed$, y el resultado de esta operación es reducido a *8-bytes* aplicando un XOR de los dos últimas sub-cadenas de *8-bytes*.

En el mecanismo de autenticación S/Key la frase secreta es memorizada por el *Usuario* y mantenida en secreto por este; mientras que el *Sistema* solo deberá almacenar el *valor hash* de la próxima contraseña, no almacenando así ningún secreto compartido en claro. Esto último es ventajoso para seguridad.

Como resultado de todas sus bondades, el mecanismo de autenticación S/Key es un algoritmo registrado, y para una mejor descripción del mismo puede consultarse el RFC 1760 del IETF (Haller, 1995). Sin embargo, este mecanismo no proporciona autenticación mutua, ya que sólo permite al *Usuario* ser autenticado pero no provee una forma de que este corrobore la identidad del *Sistema*; por tanto S/Key es vulnerable a ataques de suplantación de identidad del *Sistema* o *Servidor* (*server spoofing*).

Para solucionar la vulnerabilidad mencionada del mecanismo S/KEY, en el 2002, Yeh et al. propusieron un esquema de autenticación mediante OTPs usando tarjetas

inteligentes (Yeh, Shen, & Hwang, 2002). Esta mejora incorpora un desafío en forma de número aleatorio para proporcionar la oportunidad al *Usuario* de autenticar al *Servidor*, y brindar así protección contra ataques de tipo *server spoofing*.

En la *fase de registro* de dicho esquema, el *Servidor* le proporciona una tarjeta inteligente al *Usuario* que contiene una semilla secreta compartida $Seed$; además envía el número máximo de inicios de sesión correspondiente a dicha etapa, un número aleatorio R protegido por la semilla secreta y el *valor hash* de dicho número, como mecanismo de autenticación de mensaje (Cuadro 17). El número aleatorio R servirá para que el *Usuario* pueda verificar que se está comunicando con el *Servidor* que le ha proporcionado la semilla secreta por medio de la tarjeta inteligente. Al recibir estos valores, el *Usuario* calcula el primer valor P_0 de la *cadena de hash* a partir de su contraseña secreta y lo envía al *Servidor* protegido por un cifrado XOR con el número R. Este valor es recuperado por el *Servidor* y almacenado para el primer inicio de sesión basado en la *cadena de hash*.

Cuadro 17: Esquema del intercambio de información en la fase de registro del esquema de Yeh et al. luego del envío de la tarjeta inteligente.

$R1.\ Servidor \longrightarrow Usuario: N,\ Seed \oplus R,\ H(R)$
$R2.\ Usuario \longrightarrow Servidor: P_0 \oplus R$

Para el $i-esimo$ inicio de sesión el *Usuario* envía una solicitud al *Servidor*, a la cual este responde con los valores C_i, $Seed \oplus R_i$, $P_{i-1} \oplus H(R_i)$, basándose en el valor P_{n-1} almacenado, donde $C_i = N - i$ y R_i es no repetitivo. Al recibo de estos valores el *Usuario* obtiene el valor actual de R_i usando la semilla secreta $Seed$ y le envía el valor $P_i \oplus R_i$ al *Servidor* para su autenticación (Cuadro 18).

Cuadro 18: Esquema del intercambio de información en la $i-esima$ fase de inicio de sesión de Yeh et al.

$L1.\ Servidor \longrightarrow Usuario: C_i,\ Seed \oplus R_i,\ P_{i-1} \oplus H(R_i)$
$L2.\ Usuario \longrightarrow Servidor: P_i \oplus R_i$

Para autenticar al *Usuario*, el *Servidor* recupera P_i usando su aleatorio R_i, y calcula $H(P_i)$, para luego compararlo con el valor P_{i-1} almacenado localmente. Sólo si estos valores coinciden, el *Servidor* autentica positivamente al *Usuario*. Finalmente, el *Servidor* remplaza P_{i-1} por P_i y actualiza el valor del contador descendente n en su base de datos.

El aporte más significativo realizado por el protocolo de Yeh et al. (2002) es la combinación de la *cadena de hash* con el mecanismo *desafío-respuesta* usando el *nonce* aleatorio R. Esto resuelve básicamente la vulnerabilidad de la posible suplantación del *Servidor*; sin embargo, el *Usuario* corre con la mayor parte de costo computacional teniendo que calcular un alto número de operaciones *hash* repetidas; lo cual resulta poco factible para dispositivos con limitada capacidad cálculo o no constantemente alimentados de energía. Por otra parte, se limita el número de autenticaciones con una misma contraseña o semilla secreta. Además de ello, Tsuji y Shimizu luego demostraron que este esquema es vulnerable a ataques con robo de verificador (Tsuji & Shimizu, 2002a). En el ataque propuesto por estos autores el adversario roba el valor P_{i-1} almacenado por el *Servidor* y luego observa las comunicaciones en ambas direcciones. Con ello logra obtener la semilla secreta $Seed$ para suplantar la identidad del *Servidor*, e incluso obtiene un verificador permitiría adivinar la contraseña secreta del *Usuario*.

(b) Autenticación robusta mediante OTPs usando tarjetas inteligentes

A partir del comienzo del siglo XXI la diversificación de las aplicaciones de los protocolos de autenticación robusta usando OTPs para la variedad de redes en desarrollo, impulsaron notablemente las investigaciones en esta área, en función de proponer protocolos más completos que resolvieran las vulnerabilidades e inconvenientes de los esquemas existentes. Por otra parte, la fácil portabilidad de las tarjetas inteligentes, conjugado con el hecho de que estas pueden ser dedicadas al manejo de credenciales de autenticación e información de usuario y no están siempre conectadas a una red, impulsó el desarrollo los protocolos de autenticación de este tipo que se apoyan en las mismas.

Varias de las propuestas subsiguientes se basaron en el esquema de Yeh et al. (2002) ya analizado; en estas continuaron introduciéndose variaciones importantes al esquema original de las *cadenas de hash* inicialmente propuesto por Lamport en busca de reducir el costo computacional y brindar más robustez a la autenticación.

Lin y Hang propusieron un esquema de autenticación robusta basado en OTPs para comunicaciones móviles (M.H. Lin & Hang, 2004), que constituye una mejora del propuesto por Yeh et al. De igual forma, en esta propuesta, se utiliza una tarjeta inteligente que el *Servidor* proporciona al *Usuario* con una semilla secreta compartida, pero esta vez se reduce al mínimo la cantidad de operaciones que debe realizar el terminal del *Usuario*, pasándole al *Servidor* la responsabilidad de calcular los valores de la *cadena de hash*. Además de ello se agrega una marca de tiempo al número aleatorio propuesto por Yeh et al., buscando más robustez para el mecanismo de autenticación. En cada inicio de sesión se usan tres valores consecutivos de la *cadena de hash*, uno para que el *Usuario* verifique la identidad del *Servidor*, otro para que el *Servidor* autentique al *Usuario*, y finalmente un valor que el *Servidor* le envía por adelantado al cliente para el próximo inicio de sesión (Cuadro 19). Estos valores se envían protegidos por el *nonce* $SK_i = R_i \parallel T$ que constituye la concatenación del número aleatorio no repetitivo R_i con una marca de tiempo. El *Usuario* debe almacenar los dos valores consecutivos usados en cada fase de inicio de sesión, mientras el *Servidor* puede escoger entre almacenar solo uno de ellos y calcular dos, o almacenar dos y calcular solo uno de estos valores. La *cadena de hash* es generada a partir de un valor IK que es una función de la contraseña de Usuario, de modo que $P_i = H^{N-i}(IK)$.

Cuadro 19: Esquema del intercambio de información en la $i-esima$ fase de inicio de sesión de Lin y Hang

$L1.\ Servidor \longrightarrow Usuario: P_{i-1} \oplus SK_i, Seed \oplus SK_i$
$L2.\ Usuario \longrightarrow Servidor: P_i \oplus SK_i$
$L3.\ Servidor \longrightarrow Usuario: P_{i+1} \oplus SK_i$

Este esquema no requiere realizar cálculos de *hash* repetidas veces en cada inicio de sesión por parte del *Usuario*, pero sí por parte del *Servidor*, y aunque se reduce así el

costo computacional, se agrega un mensaje más a la etapa de inicio de sesiones. De igual forma, el número de inicios de sesión a partir de una misma semilla pre-compartida sigue siendo limitado, y se necesita sincronización de tiempo, lo cual tiene sus inconvenientes en caso de pérdida de sincronía. Por otra parte Wu et al. han probado que este esquema es vulnerable a *guessing attacks* (Wu, Liu, & Chiou, 2005), con el cual se puede intentar adivinar el número máximo N de inicios de sesión para obtener la semilla secreta $Seed$ en la fase de registro; y no se resuelve la vulnerabilidad del esquema de Yeh et al. ante el robo de verificador.

Otro método fue propuesto por Chang et al. en el 2004 (Chang, Chang, & Kuo, 2004). En este caso se elimina la *cadena de hash* y se reemplaza por un mecanismo de *desafío-respuesta* basado en un *nonce* aleatorio R_i, que se usa para generar la secuencia de OTPs, lo cual se realiza con la ayuda de dos funciones *hash* distintas $F(\cdot)$ y $H(\cdot)$. Estos valores OTPs son calculados usando la expresión $P_i = H^{F(R_i)}(K \oplus Seed)$ donde K es una función de la contraseña secreta del *Usuario*. Este esquema es resistente a los ataques a los protocolos anteriormente vistos; sin embargo, el costo computacional del mismo es incluso más elevado que todos los anteriores y también se intercambia mucha información para el inicio de sesión.

Cuadro 20: Esquema del intercambio de información en la $i-esima$ fase de inicio de sesión de Chang et al.

$L1.\ Servidor \longrightarrow Usuario: T,\ Seed \oplus R_i,\ F(R_{i-1}),\ H^2(R_i) \oplus H^2(P_{i-1} \parallel T)$
$L2.\ Usuario \longrightarrow Servidor: H^2(P_i) \oplus H(P_{i-1}),\ H(P_i) \oplus H(D_i)$

Aunque con la eliminación de la *cadena de hash* desaparece la limitación del número máximo de inicio de sesiones, en este caso también se usan marcas de tiempo. Las marcas de tiempo son usadas en varios protocolos como un mecanismo que proporciona más robustez, brindando más resistencia a *replay attack*, pero ello requiere sincronización de tiempo, lo cual actualmente es bastante evitado por los inconvenientes ya planteados en el epígrafe 2.4.

No obstante todo esto, la idea de la eliminación de la *cadena de hash* en este mecanismo, sirvió como motivación para la propuesta de este trabajo de tesis donde uno de nuestros aportes consistirá en reemplazar la *cadena de hash* en una de las fases de un protocolo actual de autenticación robusta, por otro mecanismo que resulte más eficiente, sin pérdida de seguridad para el esquema.

Lee y Chen propusieron una mejora al mecanismo de Yeh el al. que brinda protección contra el robo del *verificador*, sin degradar la eficiencia del mecanismo original (N. Y. Lee & Chen, 2005). Esta vez se protege la semilla $Seed$ mediante el *hash* de una operación XOR con el contador decreciente, y se usa un valor de desafío compuesto $SK_i = R_i \parallel T$ como en el mecanismo planteado por Lin y Hang (M.-H. Lin & Hang, 2004).

Cuadro 21: Esquema del intercambio de información en la $i-esima$ fase de inicio de sesión de Lee y Chen (N. Y. Lee & Chen, 2005)

$L1.\ Servidor \longrightarrow Usuario: R_i,\ H(Seed \oplus R_i) \oplus SK_i,\ H(SK_i) \oplus P_{i-1}$

$L2.\ Usuario \longrightarrow Servidor: P_i \oplus SK_i$

Aunque este esquema alcanza un nivel bastante satisfactorio para el momento, continúa manteniendo la sincronización de tiempo, y no resulta seguro ante la variedad de ataques sofisticados que son factibles hoy día contra esquemas de esta clase.

La creciente participación de los dispositivos inalámbricos conjugado con el aumento en la complejidad de los protocolos de autenticación robusta conllevó a una preocupación cada vez mayor por aligerar los mismos para su uso práctico. En busca de nuevas variantes para cumplir este objetivo, y de forma paralela, desde el 2000 comenzó a aparecer toda una familia de protocolos de autenticación robusta mediante OTPs que no usaban *cadenas de hash*, cuyo iniciador fue *SAS* (*Simple And Secure password authentication protocol*). La idea original fue introducida por M. Sandirigama et al. (Sandirigama, Shimizu, & Noda, 2000), cambiando totalmente la idea predominante de las *cadenas de hash*, por un mecanismo basado en *nonce* aleatorios. Sin embargo, el esquema inicial contenía varias fallas de seguridad importantes ya que, resultó vulnerable a *replay attack, password guessing attack* y ataques de negación de servicio (C. Lin, Sun,

& Hwang, 2001). Estos ataques son posibles debido a que no se verifica la integridad del valor que contiene el cambio de verificador para la próxima autenticación y a un mal uso del *nonce* aleatorio. C. L. Lin et al. proponen entonces el protocolo OSPA (*Optimal Strong-Password Authentication*), donde tratan de corregir el problema de seguridad que causa el mecanismo de cambio de verificador en el servidor (C. Lin et al., 2001); sin embargo, no lo hacen correctamente, por lo que este esquema también contiene importantes fallas de seguridad, siendo vulnerable a ataques de robo de verificador (*stolen-verifier attack*); incluso en (Tsuji & Shimizu, 2002b) se realiza un ataque de suplantación de identidad exitoso contra OSPA. Luego comienzan a aparecer versiones revisadas de SAS, en las cuales se corrige en primer lugar el envío del *nonce* aleatorio en claro, para evitar ataques de *password guessing*. En todas estas versiones se usan dos verificadores en la autenticación; el verificador actual que almacena el servidor es usado para la autenticación del *Servidor*, mientras que el verificador consecutivo se usa para enmascarar la información de autenticación y reemplazar el verificador actual una vez realizada la autenticación.

La primera versión revisada (Kamioka & Shimizu, 2001), SAS-1, combina los dos verificadores de una mejor forma. En este caso la $i-$ é$sima$ autenticación del *Usuario* es realizada por el *Servidor* comprobando que se cumple la igualdad

$$H(H(E_i) \oplus \alpha) = E_i \oplus \beta \quad (5)$$

donde $H(E_i)$ es el verificador actual almacenado en ese momento, $\alpha = H(E_i) \oplus H(E_{i+1})$, $\beta = E_i \oplus H^2(E_{i+1})$ y $E_i = H(P \oplus N_i)$; siendo P la contraseña de usuario y N_i un *nonce* aleatorio. Notemos que en la igualdad 5, dados los valores de α y β, no es factible computacionalmente encontrar un valor de E_i que la satisfaga, pues esto contradeciría la propiedad de irreversibilidad de la *función hash* segura $H(\cdot)$. El atacante no podrá entonces encontrar un valor adecuado X para remplazar a E_i, que le permita modificar α o β para cambiar el verificador actual a su conveniencia, como ocurre en el protocolo SAS original. De esta forma el protocolo SAS-1 proporciona integridad de los valores enviados, y brinda una mejor protección contra los ataques de *password guessing*, suplantación de identidad del usuario y negación de servicios. Sin embargo, Chen et al.

(C.-M. Chen & Ku, 2002) han probado que tanto este esquema como el OSPA y los anteriores son vulnerables a ataques con robo del verificador (*stolen-verifier attack*).

Cuadro 22: Esquema del intercambio de información en la $i-esima$ fase de autenticación de SAS-1.

$L1.\ Usuario \longrightarrow\ Servidor:\ ID,\ H(E_i) \oplus H(E_{i+1}),\ E_i \oplus H^2(E_{i+1})$

En el 2002 aparece la segunda versión SAS-2 propuesta por Tsuji et al., con la cual se busca reducir aún más la sobrecarga computacional, cambiando uno de los valores usados en SAS-1 por una combinación aritmética de dos valores cuyo uso no puede ser descartado (Tsuji, Kamioka, & Shimizu, 2002). El principal problema de los esquemas basados en SAS, hasta el momento, había sido la falta de autenticación mutua. Los ataques de suplantación de la identidad del servidor son una vulnerabilidad peligrosa que se han presentada en diversos reportes (T. Chen, Lee, & Horng, 2004; Stebila, Udupi, & Sheueling, 2010), por lo que SAS-2 incorpora un mecanismo de autenticación del *Servidor* para conseguir la autenticación mutua. Esto último es el aspecto más importante de SAS-2. En esta nueva versión de SAS se usan dos funciones *hash* distintas $F(\cdot)$ y $H(\cdot)$, la primera en la autenticación del *Usuario* y la segunda para autenticar al *Servidor*. De forma similar a SAS-1, para autenticar al *Usuario*, el *Servidor* usa el verificador actual almacenado y la información de autenticación recibida para obtener un valor V y un código *hash* $F(V')$, y luego verifica si se cumple cierta igualdad que involucra a dichos valores. En esta ocasión la igualdad a verificar queda del siguiente modo:

$$F(ID \parallel (E_i \oplus \beta + E_i) \oplus \alpha) = E_i \oplus \beta \quad (6)$$

donde $\alpha = E_{i+1} \oplus (E_{i+1}^2 + E_i)$, $\beta = E_{i+1}^2 \oplus E_i$, $E_i = F(ID \parallel P \oplus N_i)$ es el verificador almacenado para la $i-esima$ autenticación y $E_{i+1}^2 = F(ID \parallel E_{i+1})$.

Cuadro 23: Esquema del intercambio de información en la $i-esima$ fase de autenticación de SAS-2.

$L1.\ Usuario \longrightarrow\ Servidor:\ ID,\ E_{i+1} \oplus (E_{i+1}^2 + E_i),\ E_{i+1}^2 \oplus E_i$
$L2.\ Servidor \longrightarrow\ Usuario:\ H(ID \parallel E_{i+1}^2)$

Una vez autenticado el *Usuario*, el *Servidor* responde con el valor $H(ID \parallel E_{i+1}^2)$ usando la *función hash* $H(\cdot)$, el cual puede ser verificado por el *Usuario* usando el valor aleatorio E_{i+1} almacenado. El *Servidor* es autenticado sobre la base de que este no habría podido obtener E_{i+1}^2 correctamente sin el verificador actual E_i que es almacenado y mantenido en secreto. A pesar de las ventajas introducidas en SAS-2, dicho esquema es también inseguro contra ataques con robo de verificador, como mismo ocurre con sus versiones anteriores. Por otra parte, en ninguna de estas versiones mencionadas hasta ahora se utiliza el *nonce* aleatorio de la forma más apropiada y se usan dos verificadores en cada autenticación. Debido a estas fallas en los esquemas basados en SAS, la idea de las *cadenas de hash* se ha mantenido como la base predominante para los esquemas de autenticación robusta actuales.

(c) Autenticación robusta para el acceso a redes digitales en el hogar mediante OTPs usando tarjetas inteligentes

La computación ubicua ha tomado un papel importante en el progreso de las tecnologías y las redes de hoy día. Las redes digitales para el hogar, constituyen un nuevo paradigma de redes de área local, que permite la integración de diferentes servicios asociados con dispositivos y equipos del hogar, usando un sistema común de comunicaciones. Estas permiten conectividad a usuarios residenciales, a través de internet, para el acceso y el control de dispositivos digitales interconectados entre sí y asociados a la vida en el hogar, con un alto grado de funcionalidad. De este modo, las redes digitales en el hogar son una de las tecnologías representativas del fenómeno de la computación ubicua. Actualmente existe variedad de diseños de las mismas para el confort, ocio y cuidado de la salud de sus moradores; aunque también han sido validadas para proteger lugares y para optimizar procesos como el consumo de energía (Chan, Estève, Escriba, & Campo, 2008; Raisul, Ibne, & Mohd, 2012).

Aún cuando las redes digitales proveen conveniencias a los usuarios residenciales, proveyendo servicios de valor agregado; las características heterogéneas de las mismas, y

lo delicado del ambiente, hacen que resulte un desafío importante la protección de la privacidad y confidencialidad de la información que se transmite. Dado que es deseable que los usuarios legítimos puedan realizar el acceso remoto a distintos servicios que estas redes brindan, es necesario algún mecanismo de control de acceso a la red. Ya que, si estas redes no están bien protegidas, usuarios ilegítimos podrían conseguir acceder a estos servicios, poniendo en riesgo la seguridad en el hogar. De este modo, la autenticación de usuarios es uno de los mecanismos de seguridad más importantes requeridos en este tipo de redes, el cual ha generado creciente interés de la comunidad científica. Los mecanismos de autenticación robusta actualmente existentes, en particular los mecanismos de autenticación mediante doble factor, vienen a proporcionar una solución viable este tipo de redes, en particular aquellos basados en los esquemas *One Time Password* usando tarjetas inteligentes. Actualmente existen mecanismos bastante eficientes y seguros de OTPs, como lo son el HOTP y otros, que son factibles para su aplicación en estos escenarios en los que usualmente participan dispositivos inalámbricos.

En el 2005, Jo y Youn propusieron un protocolo de autenticación para redes residenciales basado fundamentalmente en las ideas de los esquemas de SAS, pero de forma mejorada (H.S. Jo & Youn, 2005). En el mismo cada usuario posee una tarjeta inteligente (*smart card*) firmada por el centro emisor al que está asociado el servidor de autenticación. Cada tarjeta inteligente contiene una clave privada K, un número aleatorio $Seed$ y un identificador de usuario ID. Las funciones de seguridad son ejecutadas por el punto de acceso de la red digital en el hogar (*Home Gateway*), el cual realiza el control de acceso a los recursos y servicios de la red, reemplazando el papel que hasta ahora había jugado el *Servidor* para el proceso de autenticación en los protocolos mencionados. Este esquema emplea un mecanismo *three-way challenge-response handshake* para lograr autenticación mutua. Sin embargo, usa sincronización de tiempo y el *Usuario* no usa una contraseña combinada con la tarjeta inteligente. Esto último, aunque podría parecer una ventaja no lo es, ya que investigaciones tales como (Kocher et al., 1999; Messerges et al., 2002), indican que algunos métodos sofisticados pueden extraer secretos y valores de verificación de las tarjetas inteligentes. Por lo tanto, una debilidad de los esquemas de autenticación usando tarjetas inteligentes, que requiere especial atención, radica en la

posibilidad de que la información guardada en la misma pueda ser utilizada por un adversario para construir un mensaje válido de autenticación que le permita acceder ilegalmente al *Sistema*, suplantando al *Usuario* legítimo. Sin la combinación del la tarjeta inteligente con un secreto que no pueda ser extraído de la misma, como es una contraseña de usuario, el protocolo de Jo y Youn (H.S. Jo & Youn, 2005) es totalmente vulnerable a los ataques con robo de tarjetas, los cuales se han convertido hoy día en una de las mayores preocupaciones para la seguridad de este tipo de esquemas.

Lee et al. (2005) proponen un esquema de autenticación robusta usando tarjetas inteligentes para el acceso remoto a un *Sistema*, en el cual el *Servidor* posee una clave secreta x que constituye un secreto fuerte (ya que es un número aleatorio), y el *Usuario* utiliza la tarjeta inteligente junto con una contraseña PW de libre elección para autenticarse. (N. Lee & Chen, 2005). La tarjeta inteligente almacena en este caso un valor que constituye un XOR entre la contraseña de usuario y el valor $H(ID \oplus x)$, que depende de la clave secreta x del *Servidor*. De este modo, el *Usuario* puede extraer este valor secreto haciendo uso de su contraseña y la tarjeta inteligente. El *Servidor* autentica al *Usuario* verificando que este ha obtenido dicho valor secreto correctamente. Una de las ventajas es que, en este caso, no es necesario que el *Servidor* guarde un verificador de la contraseña; pero se usan estampas de tiempo para evitar *replay attacks* y la autenticación del *Servidor* se basa en una información similar a la del *Usuario*, por lo que este esquema es vulnerable a ataques de suplantación de identidad del servidor tal y como han demostrado Yoon y Yo (2005). Por otra parte, Lee et al. también proponen un mecanismo de cambio de contraseña en la cual solo es necesario estar en posesión de la tarjeta inteligente para cambiar la contraseña sin conocer la contraseña original, lo que es vulnerable a ataques bastante simples con robo de la tarjeta inteligente (Yoon et al., 2005).

Yoon et al. tratan de corregir las vulnerabilidades del esquema de Lee et al. introduciendo un valor $K = H(ID \oplus PW \oplus x)$ usado como máscara para componer dos verificadores; uno para la contraseña de usuario ($R = K \oplus PW$), que es almacenado en la tarjeta inteligente del *Usuario* junto con K; y el otro para la clave secreta del *Servidor* ($V = K \oplus x$), que es almacenado por el Servidor. Esta vez la tarjeta inteligente verifica la

contraseña *PW* antes de generar el valor de autenticación que será enviado al *Servidor*, y si la contraseña no es correcta la autenticación es inmediatamente fallida. Dicha adaptación se incorpora también como una modificación al mecanismo de cambio de contraseña de Lee et al.; con lo que se logra evitar la vulnerabilidad de esta fase a ataques con robo de la tarjeta inteligente. Sin embargo, la fase de autenticación es también vulnerable a este tipo de ataques, ya que una vez extraído el valor *K* de la tarjeta inteligente, se puede llevar a cabo un exitoso ataque de suplantación de identidad del usuario.

Los esquemas mencionados resuelven en cierta medida el problema de la autenticación de usuarios en el ámbito de las redes digitales en el hogar, sin embargo, no incluyen ningún punto en el que se lidie de alguna forma con otras problemáticas de seguridad importantes para este tipo de escenarios, como es el caso de la privacidad de la información de contenido y el intercambio seguro de claves. Más aún, en ninguno de los mismos se brinda una descripción aceptable de la infraestructura de red para las comunicaciones.

Jeong, Chung y Choo (2006) propusieron un esquema para el acceso remoto a redes residenciales, en el que adoptan un sistema de clave pública e incorporan un *Servidor AAA (Autentication, Autorization and Acconting)* como una tercera parte confiable. La limitada capacidad de cómputo de los dispositivos inalámbricos y la necesidad de estos de ahorrar energía, hace que este tipo de soluciones, que hacen uso de criptografía de clave pública, no sean adecuadas para el entorno en cuestión. Por otra parte, el esquema es vulnerable a ataques de diccionario (*dictionary attacks*) en modo *off-line*, ya que en el primer paso del protocolo el *Usuario* le envía al *Servidor AAA*, un *hash* de su contraseña encriptado con la clave pública del *Servidor*. No obstante, la estructura de red utilizada en este esquema quedó instaurada para los esquemas posteriores como la más conveniente, a fin de cubrir de forma más integral los aspectos de seguridad requeridos para el acceso remoto a las redes digitales en el hogar.

En el 2008, Jeong, Chung y Choo proponen un esquema de autenticación de usuarios basado en OTPs usando tarjetas inteligentes, para redes residenciales, más sofisticado que el inicialmente propuesto por estos mismos autores en el 2006 (J. Jeong, Chung, & Choo,

2006). Esta vez los autores logran aligerar el protocolo propuesto utilizando criptografía de clave simétrica en vez de criptografía de clave pública. Además incorporan una fase de acceso al servicio donde se usa una clave de sesión. Sin embargo, en este esquema el *Servidor* almacena un verificador de la contraseña, y es susceptible a ataques de suplantación de identidad (*masquerading attack*) si el adversario obtiene dicho valor.

A pesar de los esfuerzos hasta este punto, todos estos esquemas aún poseen una serie de fallas de seguridad, incluyendo el hecho de que son vulnerables a los ataques con robo de la tarjeta inteligente.

Recientemente Vaidya et al. (2011) propusieron un esquema de autenticación de usuarios basado en contraseñas fuertes para proveer acceso remoto seguro en el escenario de las redes digitales en el hogar, el cual usa el mecanismo de OTP basado en HMAC, conocido como HOTP, y la técnica de las *cadenas de hash*, junto con una tarjeta inteligente. Dicho esquema está concebido no solo para resistir los ataques con pérdida de la tarjeta inteligente, sino además para proporcionar autenticación mutua, evitar la sincronización basada en tiempo y descartar el uso de verificadores de contraseña en el servidor remoto.

A continuación pasaremos a explicar levemente la infraestructura de red sobre la que se soporta este protocolo de Vaidya et al. (2011) para luego realizar un análisis detallado del mismo.

3.2 Infraestructura de red para el acceso remoto a redes digitales en el hogar

La infraestructura para la autenticación en una red digital típica para el hogar suele estar compuesta por: un punto de acceso a la red (*Home Gateway*), electrodomésticos, ordenadores, dispositivos móviles, un servidor remoto de autenticación (IAS) y un proveedor de servicios (ver Fig. 2). Dispositivos inalámbricos y móviles son utilizados por los usuarios residenciales, para conectarse al punto de acceso de la red y controlar los aparatos electrodomésticos.

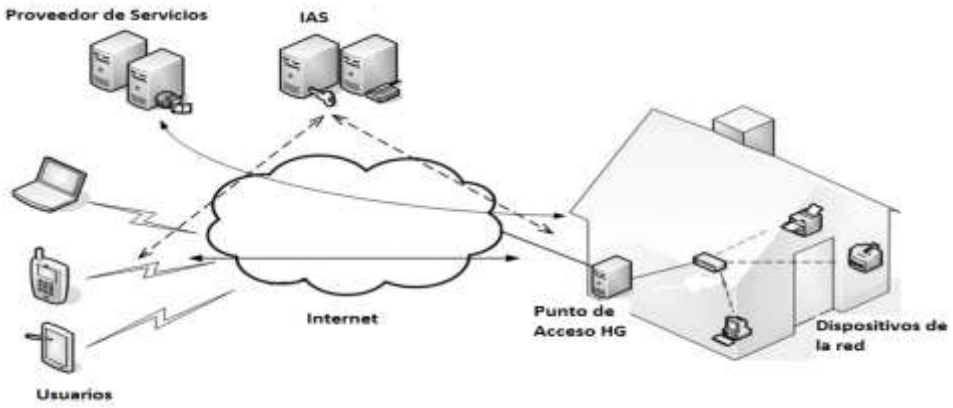


Figura 2: Arquitectura de una red digital en el hogar, (adaptada de Vaidya et al., 2011.).

Los orígenes de este tipo de modelo tripartito para la autenticación surgen con el protocolo Kerberos (Kohl & Neuman, 1991), el cual fue desarrollado por el MIT (*Massachusetts Institute of Technology*) a finales de la década de 1980, para proteger la emergente red de servicios del Proyecto "Athena". En este se usaba un Servidor de Autenticación que funcionaba como una tercera parte de confianza, la cual permitía a los nodos de la red encargados de brindar servicios, identificar de forma segura a sus entes pares solicitantes de servicios dentro del escenario de una red insegura. Esto se realizaba por medio de un intercambio tripartito en el cual el *Usuario* legítimo, para acceder a un servicio solicitado, necesitaba un "*Tiquete*" brindado por el Servidor de Autenticación, con ciertas credenciales para la corroboración de su identidad. Este *Tiquete* contenía además una clave de cifrado para permitirle al *Usuario* comunicarse con su ente par de forma segura. Este modelo ha sido ampliamente usado desde entonces y modificado para adaptarse a las necesidades actuales de seguridad. En particular, el protocolo de Vaidya et al. (2011) utiliza la escancia de este esquema. El éxito y justificación del mismo se basa en que lo más prudente para la generación, manejo e intercambio seguro de claves es utilizar una tercera parte de confianza (*trusted therd party*-TTP) como es el caso de un Servidor.

Para la infraestructura de red, ilustrada en la Fig. 2, en la cual se basa nuestro trabajo, se tiene que el punto de acceso (HG) desempeña un papel esencial en la estructura y la seguridad de la red. El HG, por un lado, proporciona conectividad interna a los diferentes

terminales de red en el hogar, interconecta la red pública y las subredes de la red doméstica, e implementa la administración remota. Por otro lado, permite a los usuarios utilizar los servicios de valor añadido brindados por los proveedores de servicios de Internet. También proporciona funciones de los servicios de seguridad, así como autenticación de usuarios para el acceso a los servicios de la red digital.

El Servidor de Autenticación Integrado (*IAS*), fuera de la red doméstica, que interviene en la mayoría de implementaciones actuales de redes digitales para el hogar, administra algunas funciones del punto de acceso, autentica a los usuarios, otorga privilegios, y ofrece controles de contabilidad. Por último, el proveedor de servicios puede suministrar muchas clases de servicios, de valor añadido a la red digital, para el disfrute de los usuarios.

3.3 Descripción general del Protocolo de Vaidya et al. (2011)

Este protocolo funciona sobre la estructura de red ilustrada en la Fig. 2, donde intervienen tres tipos de entes principales: el/los cliente/s, un punto de acceso a la red residencial (*Home Gateway* - *HG*) y un servidor integrado de autenticación (*IAS*). Este último funciona como una tercera parte de confianza para los dos primeros entes. El mismo se encarga de manejar permisos de inicio de sesión para que los usuarios puedan acceder a los servicios de la red en el hogar a través del *HG*; a su vez el *IAS* mantiene una clave secreta compartida con el *HG*.

El protocolo propuesto por Vaidya et al. consiste en cuatro fases en su totalidad:

1. *Fase de registro*: el *Usuario* escoge su propia contraseña y recibe su tarjeta inteligente (*SC*), de forma segura, la cual contiene credenciales de autenticación.
2. *Fase de autenticación e inicio de sesión*: se realiza una autenticación mutua entre el *Servidor* (*IAS*) y el *Usuario* basándose en el algoritmo *HOTP*. En esta etapa el *Usuario* y el *Servidor* acuerdan una clave de sesión única para cada vez. Al final de este proceso el servidor *IAS* le otorga al *Usuario* un tiquete (*TKG*) para autenticarse con el punto de acceso (*HG*) y acceder a los servicios de la red. La

información de autenticación en el último paso de esta fase, y el tiquete *TKG*, se envían cifrados usando criptografía simétrica.

3. *Fase de solicitud de servicios*: se realiza una autenticación mutua entre el *Usuario* y el punto de acceso *HG* a partir de credenciales del *TKG* por medio de la técnica de las *cadenas de hash*. Finalmente se le da acceso o no al *Usuario* al los servicios solicitados.
4. *Fase de cambio de contraseña*: esta fase es invocada cada vez que el *Usuario* considera conveniente cambiar su contraseña. En la misma el *Usuario* puede realizar su cambio de contraseña a partir de su contraseña válida actual y haciendo uso de su tarjeta inteligente. En base a la contraseña actual se realiza un proceso de autenticación mutua entre el *Usuario* y el *servidor IAS*, luego se usa la contraseña nueva para actualizar los valores existentes en la tarjeta inteligente en base a esta contraseña.

Tres secretos fijos constituyen la base de la seguridad del protocolo, en conjunto con la tarjeta inteligente. Estos son: la contraseña de usuario *PW*, una clave secreta "*x"* del servidor *IAS* y una clave secreta *K*, compartida entre el *Servidor* (*IAS*) y el *Usuario* (*U*). La naturaleza distinta de cada uno de estos secretos da más fortaleza al protocolo. La contraseña de usuario PW es enviada al *Servidor* IAS por un canal seguro, durante la *fase de registro*, y mantenida en secreto por el *Usuario*. La clave "*x"* del *Servidor* es mantenida también oculta por este. Los tres secretos mencionados son combinados criptográficamente para formar tres valores, que son almacenados en la tarjeta inteligente que le es entregada al *Usuario*. Dado que ninguna de las partes tiene conocimiento explícito del secreto de su contraparte, la clave secreta compartida *K* puede ser obtenida explícitamente por ambas partes, a partir de ciertos valores criptográficos y sus secretos particulares, permitiendo así que la autenticación pueda ser llevada a cabo en base a un valor común. El *Usuario U* es autenticado en base a su tarjeta inteligente y su contraseña *PW*; y puede obtener *K* a partir del valor correcto de la contraseña de usuario (*PW*) y un valor criptográfico (k_T), almacenado en su tarjeta inteligente. El servidor *IAS* puede obtener *K* a partir de su clave secreta "*x"* y un valor (u_T) enviado por el *Usuario*.

Además de los secretos mencionados existe una clave secreta pre-compartida K_{IAS-HG} entre el servidor IAS y el punto de acceso a la red digital en el hogar HG. Sobre esta clave asumiremos que ha sido previamente establecida de forma segura, mediante algún protocolo de intercambio seguro de claves entre el Servidor de Autenticación IAS y el punto de acceso HG. De este modo esta clave simétrica es preexistente, y asociada a la infraestructura de comunicaciones sobre la cual se soporta el protocolo, por lo cual no se hará ningún cuestionamiento relacionado con la misma durante nuestros análisis del protocolo.

En la *fase de autenticación e inicio de sesión*, existe un mecanismo de sincronía *U - IAS*: un contador ascendente que controla los valores generados por el algoritmo *HOTP*. Los valores de *HOTP* dependen de *K*, *PW* y el contador ascendente mantenido por cada parte; y se usan para: autenticar al *Usuario U* al comienzo de esta fase, y obtener una clave para dicha sesión combinando el valor de *HOTP* con un número aleatorio generado por *IAS*.

La *clave de sesión* es utilizada para cifrar la respuesta de autenticación (*AuthResp*) del *IAS* y le permite al *Servidor* autenticarse con el *Usuario*. Durante el último paso de la *fase de inicio de inicio de sesión*, el *Usuario* recibe un tiquete (*TKG*) para el proceso de autenticación mutua con *HG* y el acceso a los servicios. *TKG* esta encriptado usando la clave secreta pre-compartida entre *IAS* y *HG*. Este tiquete *TKG* es enviado por el *Usuario* al *HG* en la *fase inicial de solicitud de servicios*. Tanto *AuthResp* como *TKG* contienen información simétrica compartida por *U* y *HG* para inicializar y recorrer la *cadena de hash* que usarán para la autenticación mutua durante esta fase. Cada sesión tiene un tiempo de expiración T_{exp}, después del cual es necesario volver a la *fase de inicio de sesión* para recibir una clave de sesión y un tiquete nuevo para acceder nuevamente a los servicios de la red.

El proceso de autenticación mutua *U-HG* para el acceso a servicios se realiza usando el esquema tradicional de las *cadenas de hash***,** un valor criptográfico que depende de la clave de sesión y un contador descendente. La *cadena de hash* es generada a partir de un valor que depende de la *clave de sesión* y una *semilla aleatoria* compartida entre *U* y *HG*.

3.4 Desglose detallado del Protocolo de Vaidya et al. (2011).

A continuación se muestra el protocolo de Vaidya et al. (2011) de forma detallada con el objetivo de poder analizar posteriormente los pasos que conducen a vulnerabilidades del mismo, y que ello nos permita proponer modificaciones válidas para mejorarlo. Para cada fase siempre mostraremos el intercambio de mensajes en notación compacta, y luego el desglose de las acciones de cada parte en dicha fase. Tal como hasta el momento, para identificar al *Usuario* utilizaremos siempre la letra U y el *Servidor Integrado de Autenticación* lo denotaremos por IAS.

***Fase de Registro (R)*:**

Cuadro 24: Mensajes intercambiados durante la fase de registro del protocolo de Vaidya et al.

1. $U \Longrightarrow IAS: \ ID_C \ , \ PW$
2. $IAS \Longrightarrow U: \ SC \ \{ID_c, ID_{SC}, h(\cdot), v_T, g_T, k_T, C_M\}$

Paso R1. U escoge los valores ID_C y PW, y los envía a al servidor IAS por un canal seguro de comunicaciones.

Paso R2. IAS realiza las siguientes operaciones:

- Genera un secreto K
- Calcula $\begin{cases} v_T = h\ (ID_c \oplus x)\ \oplus h(PW)\ \oplus K \\ g_T = h(ID_c \parallel x \parallel K) \oplus h(ID_c \parallel PW) \\ k_T = K \oplus H\big(PW \oplus H(PW)\big) \end{cases}$
- Almacena ID_c e ID_{sc} en su base de datos de usuarios
- Escribe $\{ID_c, ID_{SC}, h(\cdot), v_T, g_T, k_T, C_M\}$ en la tarjeta inteligente

Paso R3. La autoridad encargada del IAS hace llegar la tarjeta inteligente SC al *Usuario* U por una vía segura.

***Fase de Autenticación e Inicio de Sesión (LA)*:**

En esta fase hemos modificado ligeramente la forma original de escribir el protocolo, teniendo en cuenta que la información que se envía por una de las partes no tiene por qué

coincidir con la que se recibe, ya que nuestro atacante tiene la capacidad de modificar información y comunicarse con ambas partes. De este modo, diferenciamos un valor auténtico V del valor real que se envía en nombre de este agregándole una tilde (V').

Cuadro 25: Mensajes intercambiados durante la fase de autenticación e inicio de sesión del protocolo de Vaidya et al.

1. $U \longrightarrow IAS: ID_C, u_T, a_T, G$
2. $IAS \longrightarrow U: E_{K_i}(ID_C, ID_{IAS}, N_a, A_S, N, S), E_{K_{IAS-HG}}(ID_C, ID_{IAS}, N_a, K_i, N, S, T_{exp})$

Paso LA1. U inserta su tarjeta inteligente (SC) en su terminal, o en un lector asociado a su dispositivo personal, e ingresa los valores ID_C y PW.

Paso LA2. La tarjeta SC actúa del siguiente modo:

- Verifica ID_C. Si ID_C es idéntico al ID_C almacenado por la tarjeta SC, esta continuará con el procedimiento para iniciar sesión, de lo contrario el proceso se interrumpirá.
- Obtiene los valores $\begin{cases} K = k_T \oplus h(PW \oplus h(PW)) \\ h(ID_c \oplus x) = v_T \oplus h(PW) \oplus K \end{cases}$
- Calcula $\begin{cases} u_T = h(ID_c \oplus x) \oplus K \\ a_T = h(ID_{sc} \| K) \oplus h(ID_c \| PW) \end{cases}$
- Genera el valor actual de $HOTP\ H^i(K, C_C) = HOTP(K, C_C, h(ID_c \| PW))$
- Calcula $G = h(u_T \oplus g_T) \oplus H^i(K, C_C)$
- Incrementa su contador en 1 ($C_C := C_C + 1$)

Paso LA3. U le envía $\{ID_C, u_T, a_T, G\}$ al servidor IAS

Paso LA4. Al recibo de la información $\{ID'_C, u'_T, a'_T, G'\}$, el IAS procede de la siguiente manera:

- Verifica ID_C. Si ID_C no es un identificador de usuario válido, este continuará con el procedimiento de autenticación, de lo contrario, la solicitud de inicio de sesión será rechazada.
- Obtiene $\begin{cases} K' = u'_T \oplus h(ID_c \oplus x) \\ h(ID_C \| PW') = a'_T \oplus h(ID_{SC} \| K') \end{cases}$

- Calcula $g'_T = h(ID_c \parallel x \parallel K') \oplus h(ID_c \parallel PW')$
- Obtiene $H^i(K'', C_C) = h(u'_T \oplus g'_T) \oplus G'$
- Genera $HOTP\ H^i(K', C_S) = HOTP(K', C_S, h(ID_c \parallel PW'))$
- Compara los valores $H^i(K'', C_C) \overset{?}{=} H^i(K', C_S)$, y si estos coinciden entonces IAS incrementa su contador ($C_S := C_S + 1$), de lo contrario la autenticación es fallida y se interrumpe el proceso.
- Obtiene $K_i = H^i(K, C_S)$, usando el valor actual del contador C_S.
- Genera un número aleatorio N_a y con él calcula $S_K = h(K_i \parallel N_a)$
- Calcula $A_S = h(S_K \parallel ID_C)$
- Cifra $E_{K_i}(ID_C, ID_{IAS}, N_a, A_S, N, S)$, $E_{K_{IAS-HG}}(ID_C, ID_{IAS}, N_a, K_i, N, S, T_{exp})$

Paso LA5. IAS envía una respuesta de autenticación

$AuthResp = E_{K_i}(ID_C, ID_{IAS}, N_a, A_S, N, S)$ y un tiquete de autenticación

$TKG = E_{K_{IAS-HG}}(ID_C, ID_{IAS}, N_a, K_i, N, S, T_{exp})$, al usuario U.

Paso LA6. U opera de la siguiente forma:

- Descifra $E_{K_i}(ID_C, ID_{IAS}, N_a, A_S, N, S)$ usando $K'_i = H^i(K'', C_C)$
- Obtiene $S'_K = h(K'_i \parallel N_a)$
- Calcula $A'_S = h(S'_K \parallel ID_C)$
- Finalmente chequea si $A_S \overset{?}{=} A'_S$, en caso de que coincidan da por válido el $AuthResp$ y autentica positivamente al servidor IAS.

Para acceder a los servicios de la red, los usuarios una vez autenticados pueden solicitar estos servicios al punto de acceso HG. La fase de solicitud de acceso al servicio se describe a continuación.

***<u>Fase de Solicitud de Acceso al Servicio (SR)</u>*:**

Para acceder a los servicios de la red digital en el hogar, los usuarios autenticados deben realizar la solicitud de los mismos a al puto de acceso HG. Esta fase se ilustra a

continuación en forma detallada para la *i-esima* ejecución de la misma, dentro de una sesión en la que se ha acordado la clave S_K para el cifrado de contenido.

Cuadro 26: Mensajes intercambiados durante la i-ésima ejecución de la fase solicitud de acceso al servicio del protocolo de Vaidya et al.

0. $U \longrightarrow HG : ID_C\ ,\ E_{K_{IAS-HG}}(ID_C, ID_{IAS}, N_a, K_i, N, S, T_{exp})$ [*Solo una vez por sesión*]
1. $HG \longrightarrow U : C\ , R\ , PP_{i-1}$
2. $U \longrightarrow HG : PP_i$

Paso SR1. La primera vez, el usuario U calcula el valor de OTP inicial $P_0 = F^N(S_K \oplus S)$ de una cadena de hash y lo almacena para su uso posterior.

Paso SR2. Cuando el usuario (U) requiere acceder a servicios dentro de la red digital en el hogar, este envía al punto de acceso HG el par de valores $ID_C, E_{K_{IAS-HG}}(ID_C, ID_{IAS}, N_a, K_i, N, S, T_{exp})$

Paso SR3. El punto de acceso HG actúa del siguiente modo al recibir el par de valores de la solicitud:

- Descifra $E_{K_{IAS-HG}}(ID_C, ID_{IAS}, N_a, K_i, N, S, T_{exp})$ usando la clave K_{IAS-HG}
- Verifica el valor ID_C recibido con el valor obtenido al descifrar el TKG, si estos coinciden entonces continúa con el procedimiento, de lo contrario, niega de inmediato el acceso al servicio.
- Verifica T_{exp} para ver si ha expirado la sesión
- Obtiene $S_K = h(K_i \parallel N_a)$, lo cual es realizado solo una vez por sesión
- Calcula $P_0 = F^N(S_K \oplus S)$ [sólo la primera vez]
- Inicializa un contador decreciente $C \coloneqq N$, si es la primera solicitud de la sesión y en otro caso solo disminuye su valor actual en 1, $C \coloneqq C - 1$
- Calcula $R = h(S \oplus C \oplus S_K)$ y $PP_{i-1} = P_{i-1} \oplus h(S_K)$

Paso SR4. HG envía los valores $C\ , R\ , PP_{i-1}$ al usuario U para ser autenticado por este.

Paso SR5. El usuario U realiza las siguientes operaciones

- Calcula $R' = h(S \oplus C \oplus S_K)$ y $PP'_{i-1} = P_{i-1} \oplus h(S_K)$

- Verifica que se cumpla $R' \stackrel{?}{=} R$, $PP'_{i-1} \stackrel{?}{=} PP_{i-1}$ y si estos pares de valores coinciden procede a calcular $P_i = F^{(N-i)}(S_K \oplus S)$.
- Guarda el valor de C
- Reemplaza P_{i-1} por P_i

Paso SR6. U le envía PP_i a HG

Paso SR7. HG actúa como sigue:

- Obtiene $P_i = PP_i \oplus h(S_K)$ del mensaje recibido
- Verifica que se cumple $F(P_i) = F\left(F^{(N-i)}(S_K \oplus S)\right) = F^{(N-i+1)}(S_K \oplus S) \stackrel{?}{=} P_{i-1}$
- Si finalmente es cierto que $F(P_i) = P_{i-1}$, HG reemplaza P_{i-1} por P_i y da acceso al usuario U al servicio solicitado

***Fase de Cambio de Contraseña (P)*:**

Esta fase es invocada solamente cuando el usuario desea cambiar su contraseña.

Cuadro 27: Mensajes intercambiados durante la fase cambio de contraseña del protocolo de Vaidya et al

1. $U \longrightarrow IAS: ID_c, u_T, a_T, E_K(PW, PW_N)$
2. $IAS \longrightarrow U: E_K(ID_c, c_{T_N}, v_{T_N}, g_{T_N}, k_{T_N})$

Paso P1. El usuario U inserta su tarjeta inteligente (SC) en su terminal, o en un lector asociado a su dispositivo personal, e ingresa los valores ID_C , PW y PW_N.

Paso P2. La tarjeta inteligente SC realiza las siguientes operaciones:

- Verifica ID_C. Si ID_C es idéntico al ID_C almacenado por la tarjeta SC, esta continuará con el procedimiento, de lo contrario el proceso de cambio de contraseña se interrumpirá y será rechazada la solicitud actual.

- Obtiene los valores $\begin{cases} K = k_T \oplus h\big(PW \oplus h(PW)\big) \\ h\,(ID_c \oplus x) = v_T \oplus h(PW) \oplus K \end{cases}$
- Calcula $\begin{cases} u_T = h(ID_c \oplus x) \oplus K \\ a_T = h(ID_{sc} \parallel K) \oplus h(ID_c \parallel PW) \end{cases}$
- Cifra $E_K(PW\ ,\ PW_N)$ usando la clave K.

Paso P3. El usuario U envía los valores $\{ID_c, u_T, a_T, E_K(PW\ ,\ PW_N)\}$ al servidor IAS.

Paso P4. Al recibir $\{ID_c, u_T, a_T, E_K(PW\ ,\ PW_N)\}$ El servidor IAS procede del siguiente modo:

- Verifica ID_C. Si ID_C es válido, continúa proceso de autenticación, de lo contrario se interrumpirá y será rechazada la solicitud actual.
- Calcula $K = u_T \oplus h(ID_c \oplus x)$
- Descifra $E_K(PW\ ,\ PW_N)$ usando la clave K obtenida
- Calcula $a'_T = h(ID_{sc} \parallel K) \oplus h(ID_c \parallel PW)$
- Compara a'_T con el valor de a_T recibido ($a_T \overset{?}{=} a'_T$) y si coinciden continúa proceso de autenticación, de lo contrario se interrumpirá y será rechazada la solicitud actual.
- Calcula $\begin{cases} c_{T_N} = h(ID_c \parallel PW_N) \\ v_{T_N} = h(ID_c \oplus x) \oplus h(PW_N) \oplus K \\ g_{T_N} = h(ID_c \parallel x \parallel K) \oplus c_{T_N} \\ k_{T_N} = h\big(PW_N \oplus h(PW_N)\big) \end{cases}$
- Cifra $E_K(ID_c, c_{T_N}, v_{T_N}, g_{T_N}, k_{T_N})$

Paso P5. El servidor IAS envía el valor $E_K(ID_c, c_{T_N}, v_{T_N}, g_{T_N}, k_{T_N})$ como respuesta al usuario.

Paso P6. El usuario U procede del siguiente modo:

- Descifra $E_K(ID_c, c_{T_N}, v_{T_N}, g_{T_N}, k_{T_N})$ usando la clave K obtenida inicialmente a partir de la contraseña introducida
- Calcula $c'_{T_N} = h(ID_c \parallel PW_N)$

- Verifica si c'_T es el mismo que el valor c_T recibido ($c_T \overset{?}{=} c'_T$) y si coinciden continúa proceso de autenticación, de lo contrario se interrumpirá y será rechazada la solicitud actual
- Reemplaza v_T , g_T , k_T por los nuevos valores v_{T_N} , g_{T_N} , k_{T_N} en la tarjeta inteligente.

En el próximo capítulo realizaremos un análisis detallado de este protocolo, el cual nos servirá como base para proponer modificaciones al mismo que mejoren aspectos de seguridad y de rendimiento.

CAPÍTULO 4: RESULTADOS Y DISCUSIÓN

Una vez entendidas las particularidades más importantes de los protocolos de autenticación robusta, así como las principales herramientas criptográficas que se usan en los mismos, estamos preparados para realizar un análisis minucioso del protocolo de Vaidya et al. (2011); con el objetivo de identificar sus bondades, deficiencias de rendimiento y vulnerabilidades de seguridad más importantes.

4.1 Análisis del protocolo de Vaidya et al. (2011).

Los autores Vaidya et al. afirman que el protocolo que proponen es resistente a una lista considerable de ataques, entre ellos algunos bastante sofisticados. Aunque en el artículo (Vaidya et al., 2011) se utiliza lógica de protocolos no monotónicos (Burrows, Abadi, & Needham, 1990) para la verificación del funcionamiento del protocolo planteado, este procedimiento se realiza en base a una ejecución exitosa. Esto no proporciona una prueba exhaustiva de la seguridad de dicho protocolo, que permita descartar las innumerables formas en que un adversario pueda intervenir con éxito en el mismo. Como consecuencia de esto, veremos cómo es posible realizar una ataque contra el protocolo de Vaidya et al., con el cual es posible vulnerar la seguridad en un sentido en el cual se había planteado que dicho protocolo era seguro.

A continuación realizamos un análisis lo más completo posible del protocolo de Vaidya et al. (2011) teniendo en cuenta los 12 *aspectos para la conformación de un buen protocolo de autenticación*. Esto, junto a otras observaciones, servirá entonces como base para realizar modificaciones efectivas a este protocolo que permitan lograr los objetivos de este trabajo de tesis.

(a) Análisis integral del protocolo propuesto por Vaidya et al.

1. *Reciprocidad en la identificación*: El protocolo de Vaidya et al. provee autenticación mutua entre los entes principales activos en todas sus fases.
2. *Identificación explícita*: El identificador de usuario ID_c es siempre incluido implícitamente en todos los mensajes de autenticación del protocolo y, además de ello, es incluido de forma explícita en aquellos mensajes en los que es necesario identificar al *Usuario* antes de proceder con su autenticación. Adicionalmente el identificador de su contraparte es ocasionalmente incluido de forma implícita en estos mensajes.
3. *Propiedades de los cifrados*: En el protocolo se usan los siguientes tipos de primitivas criptográficas: cifrado *one time pad* con el operador XOR, *funciones hash seguras* y cifrado con *criptografía simétrica;* siendo las dos primeras las más utilizadas debido a que son operaciones ligeras. Las propiedades de ambas han sido estudiadas en el Capítulo 1. En particular, el uso de las *funciones hash seguras* tiene como objetivo combinar criptográficamente valores secretos, en varias formas, para obtener un nuevo valor del cual estos no puedan ser deducidos. De este modo las *funciones hash* son usadas en composición con operaciones XOR y concatenación de valores. Por otra parte, las operaciones de *cifrado-descifrado* con clave simétrica son utilizadas sólo dos veces cada una en el protocolo; y la *criptografía de clave simétrica* es usada en preferencia a la *criptografía de clave privada* para proporcionar una mejor eficiencia computacional. En ninguno de los casos se especifica qué tipo de *función hash* o qué *algoritmo de cifrado* se debe usar ya que, como se ha mencionado en el Capítulo 2, estos detalles suelen dejarse fuera del diseño de los protocolos de autenticación, dando la oportunidad a aquellos que les corresponda implementar los mismos, de escoger las funciones y algoritmos que mejor se adapten a sus objetivos. No obstante esto, es importante señalar que el espacio de claves debe ser escogido suficientemente amplio como para conseguir una seguridad computacional aceptable; y el algoritmo de cifrado debe proporcionar

confidencialidad e integridad, para evitar que sea factible, para el atacante, modificar una porción de algún mensaje a su conveniencia, sin que esto sea notado por los entes principales. Por último, señalemos que aunque en la aplicación del cifrado *one time pad* se repiten valores (o claves), los mensajes cifrados siempre tienen un aspecto aleatorio, y por tanto, no es posible realizar un análisis práctico de correlación, que permita revelar valores secretos a partir de los resultados de estas operaciones XOR.

4. *Envejecimiento de claves*: Se usa una *clave de sesión* única para cada vez y cada sesión tiene un tiempo de expiración T_{exp}. Este tiempo debe ser escogido en función de la longitud de las claves y de forma tal que garantice la seguridad computacional de las mismas. Esta medida permite además limitar el alcance nocivo en caso de comprometimiento de alguna clave de sesión.
5. *Eficiencia computacional*: El protocolo propuesto por Vaidya et al. incurre en mayor costo computacional que los esquemas representativos anteriores. En su artículo (Vaidya et al., 2011), los autores comparan el costo computacional de su protocolo con otros tres esquemas representativos (N. Lee & Chen, 2005), (Jongpil Jeong, Chung, & Choo, 2008) y (Hea Suk Jo & Youn, 2005), pero no incluyen en este análisis las *operaciones hash* que resultan de recorrer la *cadena de hash* usada en la fase de *solicitud de acceso a los servicios*. Esta *cadena de hash* se recorre en orden descendente de la cantidad de *hash* anidados, por lo que se incurre, al comienzo de la fase, en un mayor número de operaciones. A pesar de ello, se le encarga al *Usuario* la generación de estos valores; lo cual resulta un inconveniente para los dispositivos inalámbricos.

   En esta parte, hemos visto una oportunidad para reemplazar esta *cadena de hash* por otro mecanismo que disminuya el costo computacional del esquema en su totalidad, sin sacrificar la seguridad del mismo, lo cual mostraremos en detalle posteriormente.
6. *Eficiencia en las comunicaciones*: En una ejecución completa y exitosa del protocolo, se intercambian un máximo de 5 mensajes, y sólo 2 después del primer acceso al servicio dentro de una misma sesión. Esto es aceptable con respecto a

otros protocolos representativos; sin embargo, veremos que es posible reducir ligeramente la cantidad de información intercambiada en el protocolo.

7. *Almacenamiento de secretos*: Ninguna de las partes almacena verificadores de la contraseña de usuario, lo cual elimina la posibilidad de ataques con robo de verificador (*stolen-verifier attack*), que han sido de los fundamentales y más repetitivos problemas de seguridad en protocolos anteriores. Por otra parte, la clave secreta compartida *K* no es almacenada por ninguno de los entes principales; y la clave secreta ¨*x*¨ del *Servidor* es sólo mantenida por este. El *IAS* suele pertenecer al *proveedor de servicios*, por lo cual se puede asumir que el mismo está protegido tras una serie de fuertes políticas de seguridad, que permiten mantener a resguardo este valor secreto.
8. *Arquitectura de comunicaciones*: La arquitectura de comunicaciones se tiene en cuenta en el protocolo y la estructura de red subyacente está bastante bien descrita en el artículo de Vaiya et al. (2011); de esa misma forma, el papel que asume cada ente que participa en el protocolo se ha explicado.
9. *Actuación de cada principal*: La forma de actuar de cada *ente principal*, ante cada intercambio de información en el que se ve involucrado, ha sido explicada en el Capítulo 3, de la misma forma en que lo han hecho sus autores.
10. *Amigable para el usuario*: Aunque el usuario deberá portar una tarjeta inteligente, junto al dispositivo que le permita acceder a los servicios de la red digital en el hogar, se asume que en este entorno, que supone un alto nivel tecnológico, dicho elemento no constituye una dificultad, sobre todo teniendo en cuenta que las tarjetas inteligentes son de fácil portabilidad y bajo costo. Además de esto, un punto importante para la facilidad de uso del protocolo, es el hecho de que el *Usuario* puede escoger su contraseña libremente (siempre que esta sea una contraseña fuerte).
11. *Intervención de terceros*: Tal y como se ha aclarado, el *Servidor Integrado de Autenticación* funciona como una *tercera parte de confianza* para el Usuario (*U*) y el punto de acceso *HG*; y ambas partes deben confiar en que el mismo generará

claves criptográficas que sean “suficientemente buenas” para los procesos de autenticación en que son utilizadas.

12. *Identificación de sesiones*: Cada *sesión* del protocolo viene identificada por la *clave de sesión* correspondiente que depende además de los valores generados por el algoritmo *HOTP*, el cual utiliza un contador ascendente como factor de cambio (*moving factor*). Además de ello, cada mensaje de autenticación intercambiado en la *fase autenticación e inicio de sesión* es criptográficamente ligado, de alguna forma, con el valor actual de *HOTP* o con la *clave de sesión actual*. Posteriormente, en la *fase de solicitud de acceso al servicio*, cada mensaje está ligado a un valor distinto de la *cadena de hash* que corresponde a la *sesión actual*, la cual es generada a partir de la *clave de sesión actual* combinada con una *semilla aleatoria*; y se incorpora además un *contador descendente*. Estos dos últimos valores, que funcionan como *nonce*, imprimen dinamismo a los mensajes de autenticación intercambiados durante esta fase del protocolo y permiten la identificación unívoca de los mismos, con el objetivo de prevenir *replay attacks*.

Además de los aspectos analizados, se puede agregar que el protocolo de Vaidya et al. (2011) no usa sincronización de tiempo, lo que puede considerarse ventajoso. Por otra parte, el hecho de que este cuente con una *fase de cambio de contraseña* con autenticación mutua es un elemento importante para la seguridad, ya que disminuye la posibilidad de ataques de negación de servicio (*deny of service attacks*).

(b) Análisis de la seguridad del protocolo de Vaidya et al.

Antes de comenzar con el análisis de la seguridad del protocolo en cuestión recordemos con qué tipo de atacantes se supone que debemos lidiar. En estos casos es realista y conveniente asumir que el atacante tiene el control total sobre los canales de comunicación. O sea, que este puede: interceptar y almacenar todo tipo de mensajes en cualquier dirección, modificar cualquier mensaje recibido y establecer comunicaciones con cualquiera de las partes en cualquier momento. En este caso, agregaremos la

posibilidad de que el adversario pueda además obtener la tarjeta inteligente ilícitamente y extraer información de la misma. Esto último no es exagerado ya que investigaciones tales como (Kocher et al., 1999), indican que algunos métodos sofisticados pueden extraer secretos y valores de verificación de las tarjetas inteligentes. También consideraremos la posibilidad, dependiendo del caso, de que el atacante pueda obtener algún secreto almacenado por alguno de los entes principales que intervienen en el protocolo. Recordemos a la vez, que hemos asumido que ningún ente principal se comportará de forma maliciosa; por lo que descartaremos la posibilidad de ataques internos.

Teniendo en cuenta las posibilidades del atacante, es necesario verificar que la combinación de los valores almacenados en la tarjeta inteligente y los valores que forman parte de los mensajes que se intercambian, no permitan obtener alguno de los secretos básicos del protocolo.

Estos valores, hasta la culminación de la *fase de autenticación e inicio de sesión* son los siguientes:

- $v_T = h\,(ID_c \oplus x)\ \oplus h(PW)\ \oplus K$
- $g_T = h(ID_c \parallel x \parallel K) \oplus h(ID_c \parallel PW)$
- $k_T = K \oplus H\big(PW \oplus H(PW)\big)$
- $u_T = h(ID_c \oplus x) \oplus K$
- $a_T = h(ID_{sc} \parallel K) \oplus h(ID_c \parallel PW)$
- $G = h(u_T \oplus g_T) \oplus H^i(K, C_C)$
- $AuthResp = E_{K_i}(ID_C, ID_{IAS}, N_a, A_S, N, S)$
- $TKG = E_{K_{IAS-HG}}(ID_C, ID_{IAS}, N_a, K_i, N, S, T_{exp})$

Una revisión detenida de estos valores nos lleva a observar que existe una dependencia entre el valor $u_T = h(ID_c \oplus x) \oplus K$, enviado por el *Usuario* en el *paso LA3*, y el valor $v_T = h\,(ID_c \oplus x)\ \oplus h(PW)\ \oplus K$, almacenado en la tarjeta inteligente (SC). De modo que estos cumplen la relación

$$v_T = u_T \oplus h(PW) \quad (7)$$

Esto implica que se puede obtener un verificador de la contraseña a partir de u_T y v_T; dando lugar así a un *ataque de verificación de contraseñas con pérdida de la tarjeta*

inteligente (*password guessing attack with lost smart card*), tal y como se describe en (Kim & Kim., 2011).

Password guessing attack with lost smart card: Este ataque consiste en tratar de obtener la contraseña de un *Usuario* de un *Sistema*, a partir de información almacenada en la tarjeta inteligente, y haciendo uso de cualquier tipo de información recopilada por el atacante, que haya sido capturada durante las comunicaciones entre las partes principales del protocolo. Este tipo de ataque suele llevarse a cabo probando posibles valores de contraseñas de una lista dada como puede ser un *Diccionario*.

Kim et al. han planteado un ataque efectivo de este tipo contra el protocolo de Vaidya et al. (2011), el cual desglosamos detalladamente a continuación.

(c) Ataque de verificación de contraseñas con pérdida de la tarjeta inteligente contra el protocolo de Vaidya et al .(2011), adaptado de (Kim et al., 2011).

Paso1: El atacante intercepta los mensajes de una sesión del *Usuario* durante la *fase de autenticación e inicio de sesión* y logra obtener la tarjeta inteligente del *Usuario* de forma ilícita.

Paso2: El atacante ha almacenado el valor fijo $u_T = K \oplus h(ID_c \oplus x)$ obtenido de las comunicaciones interceptadas y logra extraer el valor $v_T = h\,(ID_c \oplus x)\ \oplus h(PW)\ \oplus K$ almacenado en la tarjeta inteligente.

Paso3: El atacante escoge una contraseña candidata PW' de un *Diccionario* y procede a calcular

$$\begin{aligned} u'_T = \ v_T \oplus h(PW') &= h\,(ID_c \oplus x)\ \oplus h(PW)\ \oplus K \oplus h(PW') \\ &= h\,(ID_c \oplus x) \oplus K \oplus \big(h(PW) \oplus h(PW')\big) \\ &= u_T \oplus \big(h(PW) \oplus h(PW')\big)\ . \end{aligned}$$

Donde $u'_T = u_T \iff h(PW) \oplus h(PW') = 0$, por lo que de cumplirse la primera igualdad el atacante sabrá que ha encontrado la contraseña, ya que la *función hash* usada $h(\cdot)$ es una *función segura*, y de ser $PW' \neq PW$ se habría encontrado una colisión.

Paso4: Si el valor u'_T calculado con la contraseña candidata no coincide con el valor u_T original, el atacante repite el proceso con otro valor del diccionario hasta encontrar la contraseña correcta o agotar el *Diccionario* completo.

Con esto, queda probado que el protocolo propuesto por Vaidya et al. falla en brindar seguridad contra ataques con robo de la tarjeta inteligente, contrario a lo que sus autores habían afirmado.

(d) Segundo ataque contra el protocolo de Vaidya et al.

Un segundo ataque menos nocivo es posible contra el protocolo de Vaidya et al., para obtener los valores HOTP asociados con la información de autenticación en la primera parte de la *fase de autenticación e inicio de sesión*:

Paso1: El atacante intercepta los mensajes de una sesión del *Usuario* durante la *fase de autenticación e inicio de sesión* y logra obtener la tarjeta inteligente del *Usuario* de forma ilícita.

Paso2: El atacante ha almacenado el valor fijo $u_T = K \oplus h(ID_c \oplus x)$ obtenido de las comunicaciones interceptadas, y logra extraer el valor $g_T = h(ID_c \parallel x \parallel K) \oplus h(ID_c \parallel PW)$ almacenado en la tarjeta inteligente.

Paso3: Con estos dos valores el atacante podrá calcular $h(u_T \oplus g_T)$ y con ello obtener $H^i(K, C_C) = G \oplus h(u_T \oplus g_T)$.

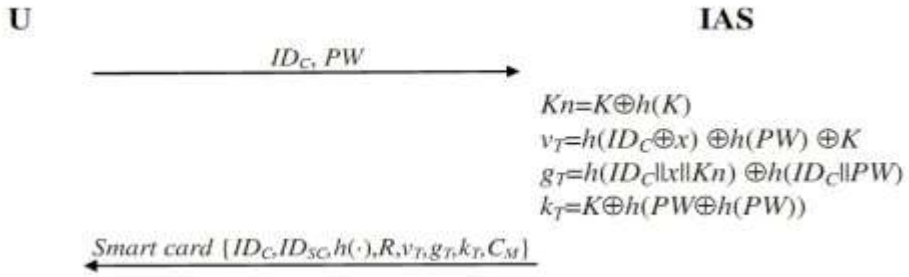
Esta posibilidad comprometería los valores K_i generados con anterioridad, lo cual permitiría descifrar el mensaje de autenticación $E_{K_i}(ID_C, ID_{IAS}, N_a, A_S, N, S)$ para obtener luego las claves de sesión anteriores $S_K = h(K_i \parallel N_a)$. De esta forma, el protocolo de Vaidya et al. no cumple con la propiedad *secret key forward secrecy*, la cual garantiza que contenidos confidenciales que hayan sido cifrados con las claves de sesión anteriores permanezcan seguros.

(e) La propuesta de Kim et al. (2011)

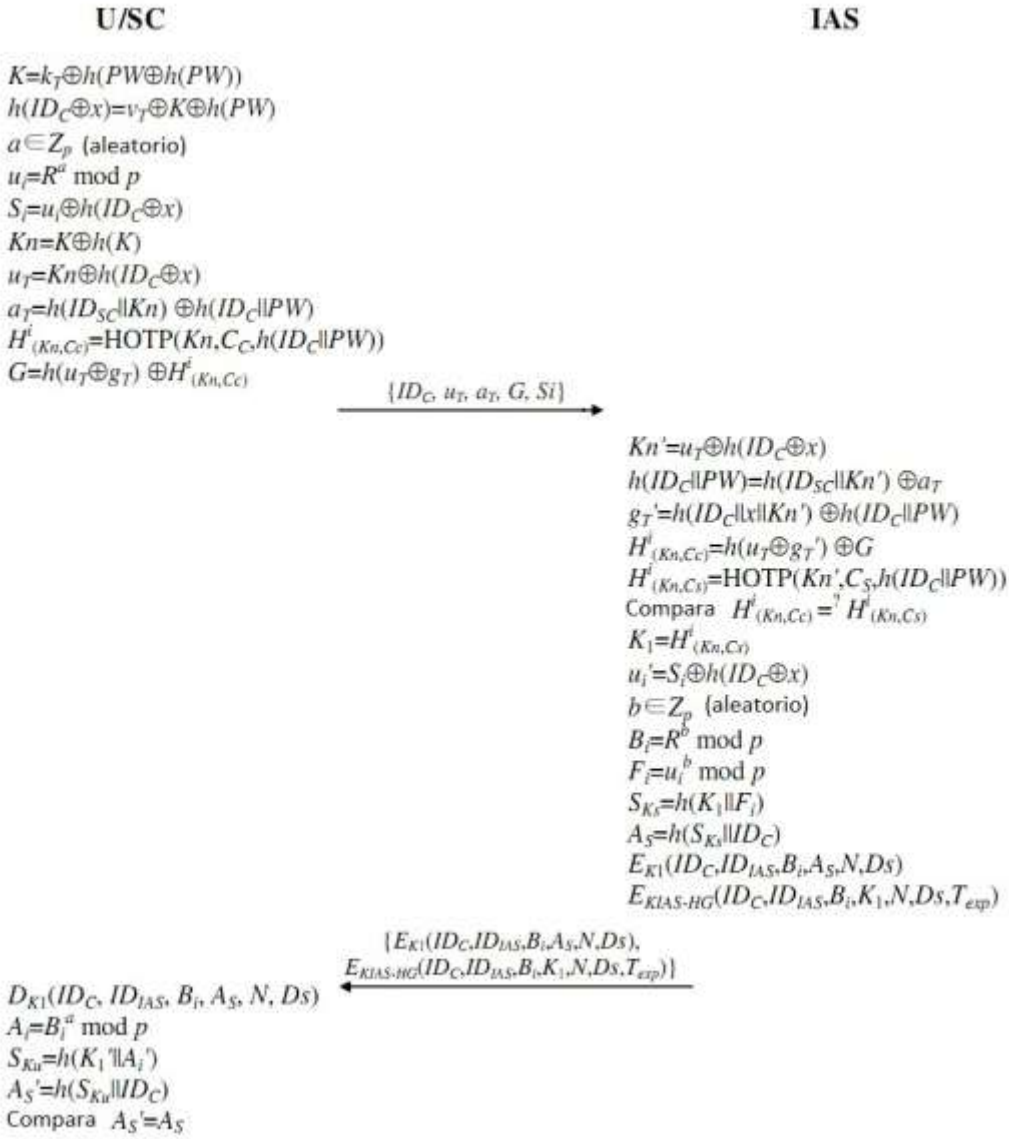
Como una alternativa para mejorar la seguridad del protocolo de Vaidya et al. frente a estas dos vulnerabilidades que hemos evidenciado, Kim et al. (2011) proponen un nuevo protocolo (AUTH HOTP) basado en el de Vaidya et al. (2011). Para lograr sus objetivos Kim et al. introducen un valor adicional $\boldsymbol{K_n} = \boldsymbol{K} \oplus h(K)$ y remplazan el valor de u_T original por $u_T = \boldsymbol{K_n} \oplus h(ID_c \oplus x)$, entre otras modificaciones basadas en el protocolo *Diffie-Helman*, agregando así otras operaciones criptográficas. Con ello logran un protocolo más seguro. Sin embargo, esta propuesta incrementa el costo computacional del protocolo original, que ya era notable para este tipo de escenarios, donde participan dispositivos inalámbricos, y también aumenta la cantidad de información que se intercambia durante la autenticación.

Las fases de *registro, autenticación e inicio de sesión* y *solicitud de acceso al servicio* del protocolo propuesto por Kim et al. pueden verse en los Cuadros 28, 29 y 30 respectivamente, que aparecen a continuación, donde están ilustradas usando notación compacta.

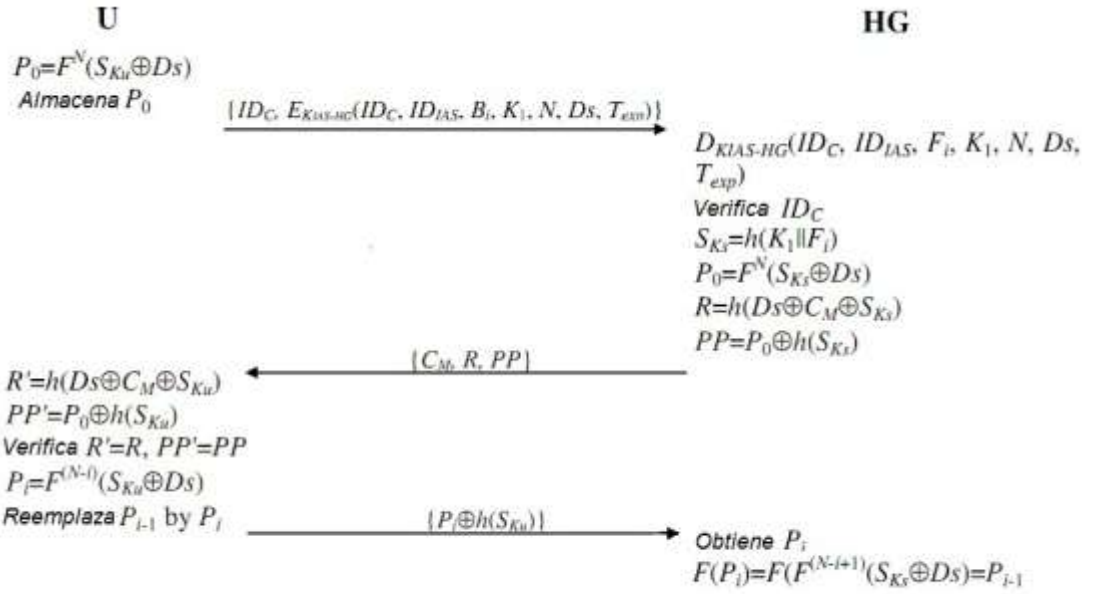
Cuadro 28: Fase de registro del protocolo de Kim et al. (2011) extraído del artículo (Kim et al., 2011).



Cuadro 29: Fase de autenticación en inicio de sesión del protocolo de Kim et al. (2011), extraído del artículo (Kim et al., 2011).



Cuadro 30: Fase de solicitud de acceso al servicio del protocolo de Kim et al. (2011), extraído del artículo (Kim et al., 2011).



Kim et al. han implementado además un ataque contra el protocolo de Vaidya et al. para el cual es necesario asumir que el atacante, además de tener el control de los canales de comunicación, de conseguir la tarjeta inteligente y de extraer lo valores de la misma, es capaz de obtener adicionalmente la clave secreta "x" del Servidor (IAS). Bajo estas condiciones Kim et al. logran probar que es posible obtener la clave de sesión en el protocolo de Vaidya et al.; sin embargo, estos tipos de ataques son bastante complejos e inusuales, por lo que ello requiere del adversario. El protocolo propuesto por Kim et al. (2011) dedica la mayoría de sus modificaciones a brindar protección contra este ataque particular; y aunque logran asegurar, hasta cierto punto, el comprometimiento de las claves de sesión, no llegan a caer en cuenta sobre el hecho de que estas medidas no libran a su protocolo de otros ataques, incluso menos complicados y más nocivos, que pueden ser llevados a cabo por un atacante con las capacidades mencionadas. A continuación mostramos un ataque que hemos elaborado contra el protocolo de Kim et al., el cual puede ser llevado a cabo por cualquier atacante con las potencialidades mencionadas anteriormente.

(f) Ataque contra el protocolo de Kim et al. con robo de la tarjeta inteligente y robo de la clave secreta del Servidor remoto de autenticación

Paso1: El atacante intercepta los mensajes de una ejecución de la fase de *autenticación e inicio de sesión* y almacena el valor $u_T = K_n \oplus h(ID_c \oplus x)$.

Paso2: El atacante en posesión de la clave secreta "x" puede actuar de la misma forma que lo hace el Servidor IAS, de modo que puede obtener el valor $K_n = u_T \oplus h(ID_c \oplus x)$ y con ello puede conseguir $h(ID_c \| PW) = h(ID_{sc} \| K_n) \oplus a_T$.

Paso3: Dado que el valor $h(ID_c \| PW)$ puede ser utilizado como un verificador de contraseña el atacante escoge una contraseña candidata PW' de un diccionario y verifica si $h(ID_c \| PW') = h(ID_c \| PW)$, repitiendo el proceso hasta que la igualdad se cumpla; en cuyo caso sabrá que ha encontrado la contraseña válida del *Usuario*.

Paso4: A partir de entonces el atacante podrá autenticarse positivamente con el IAS y recibir el tiquete de autenticación para acceder a los servicios de la red como si este fuera el *Usuario* legítimo; llevando a cabo una exitosa suplantación de identidad del *Usuario*.

De este modo, el protocolo de Kim et al. (2011) tampoco es seguro contra ataques mixtos con robo de la calve secreta "x" del *Servidor*.

4.2 El Protocolo modificado

Teniendo en cuenta todo lo analizado hasta el momento respecto al protocolo de Vaidya et al. (2011), y otras observaciones particulares que posteriormente expondremos, hemos realizado modificaciones para elevar el nivel de seguridad del mimo, simplificar ligeramente la complejidad del protocolo y mejorar su eficiencia.

(a) Análisis y justificación de las modificaciones al protocolo de Vaidya et al.(2011)

La dependencia entre los valores $v_T = h\,(ID_c \oplus x) \oplus h(PW) \oplus K$ y $u_T = K \oplus h(ID_c \oplus x)$ en el protocolo de Vaidya et al. puede ser eliminada, cambiando el primero de estos por otro valor que no permita obtener un verificador de la contraseña con ninguno de los valores transmitidos en claro o almacenados en la tarjeta inteligente. En este caso hemos sustituido v_T por $u_2 = h(ID_C \oplus x) \oplus h(ID_C \parallel PW) \oplus h(K)$. Esta sustitución permite realizar con este valor los procedimientos necesarios para el correcto funcionamiento del protocolo y elimina la posibilidad de ejecutar un *ataque de verificación de contraseñas con pérdida de la tarjeta inteligente* como el que ha sido planteado en el acápite 4.1 c); lo cual justificaremos más adelante al realizar un análisis de seguridad de nuestro protocolo.

Si se observan detenidamente las dos primeras fases del protocolo de Vaidya et al. se puede advertir que el valor g_T es sólo usado dentro de $G = h(u_T \oplus g_T) \oplus H^i(K, C_C)$ con el objetivo de enmascarar el valor HOTP, generado en cada sesión. Sin embargo, como hemos mostrado en 4.1 d), un atacante que logre obtener los valores de la tarjeta inteligente e interceptar los mensajes de una ejecución de la *fase de autenticación e inicio de sesión* podrá calcular acertadamente $h(u_T \oplus g_T)$ y, con ello, obtener el valor actual de HOTP $H^i(K, C_C)$. Esta falla de seguridad puede ser corregida eliminando el valor g_T, en el protocolo de Vaidya, y reemplazando $h(u_T \oplus g_T)$ por un *hash* de los valores $h(ID_C \oplus x)$ y $h(ID_C \parallel PW)$, si se garantiza que los mismos no puedan ser obtenidos como una combinación de los valores almacenados en la tarjeta inteligente y los enviados en claro. Esto último se logra en nuestro protocolo. De igual forma, nos hemos asegurado de que estos dos valores puedan ser obtenidos por el servidor IAS a partir de la clave secreta "x" y la información de autenticación que le es enviada por el *Usuario*, y por el *Usuario* partir de su contraseña PW y su tarjeta inteligente.

Además se puede observar que, durante la última parte de la *fase de autenticación e inicio de sesión*, Vaidya et al. no brindan integridad al valor TKG. Esto, aunque no resulta crítico para la seguridad, permite que un atacante pueda importunar al *Usuario* enviando un tiquete falso, sin que el *Usuario* pueda notarlo hasta que falle su primer intento de

acceso al servicio a través del punto de acceso HG. Una forma de evitar esto y, a la vez, proporcionar un mecanismo de autenticación para el Servidor IAS, es incluir este tiquete dentro de la información cifrada que contiene el valor *AuthResp*. De este modo se obtiene un mecanismo de autenticación de entidad y de mensaje al mismo tiempo.

La parte que consideramos más relevante en nuestras modificaciones se encuentra en la *fase de solicitud de acceso al servicio*, donde hemos reemplazado la *cadena de hash* por un mecanismo de *sincronización secuencial*, basado en el *valor de estados semi-aleatorios* E_n, el cual constituye un nuevo tipo de *nonce* presentado en esta tesis en el Capítulo 2. Esta secuencia es mantenida simultáneamente por el usuario U y el punto de acceso HG. La misma es obtenida a partir de un estado inicial compartido E_1, que es combinado progresivamente con una secuencia de números aleatorios A_n, que funcionan como desafío y aportan cierta incertidumbre al proceso de cambio de la secuencia de estados. Así, cada próximo valor E_{n+1} se obtiene en función de los valores anteriores E_n y A_n utilizando una *función hash segura*. Este tipo de secuencia aporta sincronización con alto nivel de incertidumbre y, a la vez, incorpora un *mecanismo de desafío-respuesta*, que permite una autenticación bidireccional con un solo valor de estado. De este modo, el primer sentido de la autenticación se basa en la *sincronización de los estados*, y el otro sentido puede ser completado gracias al *mecanismo de desafío-respuesta* incorporado. A diferencia de la *cadena de hash*, en la cual el número de *operaciones hash* requeridas depende de la posición de la cadena en la que se encuentren sincronizados el *Usuario* y el *Servidor*; en la secuencia utilizada, se requiere una cantidad fija de *operaciones hash*, en todo momento. En el protocolo de Vaidya et al., el cálculo de los valores de la *cadena de hash* se le encarga al *dispositivo del usuario*, lo cual atenta contra las limitadas posibilidades de cómputo y la necesidad de ahorro energético de aquellos dispositivos inalámbricos que participen en el protocolo. Los métodos introducidos en nuestro protocolo favorecen a estos dispositivos, ya que se logra una reducción considerable del costo computacional en esta fase del protocolo, como mostraremos posteriormente.

Nuestra versión del *protocolo de Vaidya et al. modificado*, se expone en detalle a continuación para cada una de las cuatro fases correspondientes con las que hemos estado

tratando. Señalemos que algunas notaciones han sido ligeramente modificadas para facilitar el análisis y la comprensión de las secuencias de pasos que se exponen.

(b) Fase de Registro modificada

Cuadro 31: Mensajes intercambiados durante la fase de registro del protocolo de Vaidya et al. modificado.

1. $U \Longrightarrow IAS: ID_C, PW$
2. $IAS \Longrightarrow U: SC\{ID_c, ID_{SC}, h(\cdot), u_1, u_2, C_M\}$

Paso R1. $\boxed{\text{U}}$ escoge ID_C , PW envía estos valores al $\boxed{\text{IAS}}$ por una vía segura.

Paso R2. $\boxed{\text{IAS}}$ realiza las siguientes operaciones:

Genera un secreto K

- Calcula $\begin{cases} u_1 = h\big(PW \oplus h(PW)\big) \oplus K \\ u_2 = h(ID_C \oplus x) \oplus h(ID_C \parallel PW) \oplus h(K) \end{cases}$
- Almacena ID_C , ID_{SC}
- Escribe $\{ID_C, ID_{SC}, h(\cdot), u_1, u_2, C_M\}$ en la tarjeta $\boxed{SC}$

Paso R3. La autoridad encargada del $\boxed{\text{IAS}}$ hace llegar la tarjeta inteligente (SC) al usuario $\boxed{U}$ por una vía segura

(c) Fase de Autenticación e Inicio de Sesión Modificada

Cuadro 32: Mensajes intercambiados durante la fase de autenticación e inicio de sesión del protocolo de Vaidya et al. modificada

1. $U \longrightarrow IAS: ID_C, s_1, s_2, G$
2. $IAS \longrightarrow U: E_{K_i}(ID_C, ID_{IAS}, TKG_C, E_1, S), E_{K_{IAS-HG}}\big(ID_C, ID_{IAS}, S_{k_i}, E_1, T_{exp}\big)$

Paso LA1. El usuario $\boxed{U}$ inserta su tarjeta inteligente en su terminal o en un lector asociado a su dispositivo personal, e ingresa los valores ID_C y PW.

Paso LA2. La tarjeta $\boxed{SC}$ actúa del siguiente modo:

- Verifica ID_C. Si ID_C es idéntico al ID_C almacenado por la tarjeta $\boxed{SC}$ esta continuará con el procedimiento para iniciar sesión, de lo contrario el proceso se interrumpirá.
- Obtiene los valores $\begin{cases} K = h\big(PW \oplus h(PW)\big) \oplus u_1 \\ h(ID_C \oplus x) = h(K) \oplus h(ID_C \| PW) \oplus u_2 \end{cases}$
- Calcula $\begin{cases} s_1 = h(ID_C \oplus x) \oplus K \\ s_2 = h(ID_C \| PW) \oplus h(ID_{SC} \| K) \end{cases}$
- Genera el valor actual de

$$HOTP\, H^i(K, 2C_C) = HOTP\big(K, 2C_C, h(ID_c \| PW)\big)$$

- Calcula $G = h\big(h(ID_C \oplus x) \| h(ID_C \| PW)\big) \oplus H^i(K, 2C_C)$

Paso LA3. $\boxed{U}$ le envía $\{ID_C, s_1, s_2, G\}$ al servidor $\boxed{\text{IAS}}$

Paso LA4. $\boxed{\text{IAS}}$ procede de la siguiente manera al recibo de la información:

- Verifica ID_C. Si ID_C no es un identificador de usuario válido, esta continuará con el procedimiento la verificación de identidad y autenticación, de lo contrario, la solicitud de inicio de sesión será rechazada.
- Obtiene $\begin{cases} K = h(ID_C \oplus x) \oplus s_1 \\ h(ID_C \| PW) = s_2 \oplus h(ID_{SC} \| K) \end{cases}$
- Calcula $h\big(h(ID_C \oplus x) \| h(ID_C \| PW)\big)$
- Obtiene $H^i(K, 2C_C) = h\big(h(ID_C \oplus x) \| h(ID_C \| PW)\big) \oplus G$
- Genera $HOTP\, H^i(K, 2C_S) = HOTP\big(K, 2C_S, h(ID_c \| PW)\big)$
- Compara los valores $H^i(K, 2C_C) \overset{?}{=} H^i(K, 2C_S)$, y si estos coinciden continua con el proceso de autenticación, de lo contrario interrumpe.
- Obtiene $K_i = H^i(K, 2C_S + 1)$
- Incrementa su contador $C_S := C_S + 1$
- Genera dos números aleatorios S y E_1
- Calcula $S_{k_i} = h(K_i \oplus S)$
- Cifra $E_{K_{IAS-HG}}\big(ID_C, ID_{IAS}, S_{k_i}, E_1, T_{exp}\big) = TKG$

- Calcula $TKG_C = h(TKG \parallel ID_C)$
- Cifra $E_{K_i}(ID_C, ID_{IAS}, TKG_C, E_1, S) = AuthResp$

Paso LA5. IAS envía una respuesta de autenticación $AuthResp$ y el tiquete de autenticación TKG al usuario U.

Paso LA6. U opera de la siguiente forma:

- Descifra $E_{K_i}(ID_C, ID_{IAS}, TKG_C, E_1, S)$ usando $K_i' = H^i(K, 2C_C + 1)$
- Verifica el valor ID_C descifrado y si es válido continúa, de lo contrario interrumpe.
- Calcula $h(TKG \parallel ID_C)$ usando el valor TKG que ha recibido en claro y lo verifica con el valor TKG_C descifrado.
- En caso de que coincidan, da por válido el $AuthResp$ y autentica positivamente al servidor IAS.
- Finalmente U incrementa su contador en 1: $C_C := C_C + 1$

(d) Fase de Solicitud de Acceso al Servicio Modificada

En esta fase se usa un mecanismo basado en la combinación de una *sincronización secuencial* combinado con *desafío-respuesta* mediante números aleatorios. En cada ejecución de esta fase, un valor de estado E_n almacenado por el *Usuario*, servirá como base para la sincronización entre este y el punto de acceso HG. El último almacenará siempre un valor de estado propio G_n que constituye un valor trampa de E_n. Cada valor E_{n+1} dependerá de los valores anteriores E_n y un *nonce* aleatorio A_n, a partir de una fórmula recurrente. El subíndice indica el número de la sesión que se ejecuta actualmente. En nuestra propuesta dicho mecanismo reemplaza a la *cadena de hash* de esta fase en esquema de Vaidya et al.

Cuadro 33: Mensajes intercambiados durante la *i-ésima* ejecución de la fase solicitud de acceso al servicio del protocolo de Vaidya et al. modificada

0. $U \longrightarrow HG : ID_C\ ,\ E_{K_{IAS-HG}}(ID_C, ID_{IAS}, S_{k_i}, E_1, T_{exp})$	[*Solo una vez por sesión*]
1. $HG \longrightarrow U : A_n \oplus h(S_{k_i})\ ,\ h(A_n \parallel G_n)$	
2. $U \longrightarrow HG : h(E_n) \oplus h(A_n)$	

Paso SR1. Cuando el usuario $\boxed{U}$ requiere acceder a servicios dentro de la red digital en el hogar, este envía al punto de acceso $\boxed{HG}$ el par de valores $ID_C,\ E_{K_{IAS-HG}}(ID_C, ID_{IAS}, S_{k_i}, E_1, T_{exp})$

Paso SR2. El punto de acceso $\boxed{HG}$ actúa del siguiente modo al recibir el par de valores de la solicitud:

- Descifra $E_{K_{IAS-HG}}(ID_C, ID_{IAS}, S_{k_i}, E_1, T_{exp})$ usando la clave K_{IAS-HG}
- Verifica el valor ID_C recibido con el valor obtenido al descifrar el TKG, si estos coinciden entonces continúa con el procedimiento, de lo contrario niega de inmediato el acceso al servicio
- Verifica T_{exp} para ver si ha expirado la sesión
- Guarda los valores $ID_C,\ ID_{IAS}, S_{k_i}, E_1, T_{exp}$
- Calcula $h(S_{k_i})$ y lo almacena para usos posteriores
- Calcula $G_1 = E_1 \oplus h(E_1)$ [sólo la primera vez]
- Genera A_n aleatorio
- Calcula $A_n \oplus h(S_{k_i})\ ,\ h(A_n \parallel G_n)$

Paso SR3. $\boxed{HG}$ envía los valores $A_n \oplus h(S_{k_i})\ ,\ h(A_n \parallel G_n)$, al usuario $\boxed{U}$ para ser autenticado por este.

Paso SR4. El usuario $\boxed{U}$ realiza las siguientes operaciones:

- Obtiene $A_n = \left(A_n \oplus h(S_{k_i})\right) \oplus h(S_{k_i})$ utilizando su valor de la clave S_{k_i}
- Calcula $G_n = E_n \oplus h(E_n)$ a partir de su propio valor de estado E_n
- Calcula $h(A_n \parallel G_n)$
- Verifica el valor $h(A_n \parallel G_n)$ recibido de $\boxed{\text{HG}}$ con el valor calculado anteriormente
- Si coinciden entonces calcula los valores $h(E_n) \oplus h(A_n)$

y $E_{n+1} = h\left(\left(E_n \oplus h(A_n)\right) \| S_{k_i}\right)$ de lo contrario se interrumpe la autenticación.

- Reemplaza su valor secuencial E_n por E_{n+1}

Paso SR6. $\boxed{U}$ le envía $h(E_n) \oplus h(A_n)$ a $\boxed{HG}$

Paso SR7. $\boxed{HG}$ actúa como sigue:

- Obtiene $h(E_n')$ a partir de A_n almacenado y el valor recibido mediante la operación $\left(h(E_n) \oplus h(A_n)\right) \oplus h(A_n)$.
- Verifica que si se cumple $h(E_n') = h(h(E_n') \oplus G_n)$ usando su valor de estado G_n, si esto se cumple entonces $E_n' = E_n$ por lo que continúa, de lo contrario interrumpe.
- Calcula $E_{n+1} = h\left(\left(h(E_n) \oplus h(A_n)\right) \| S_{k_i}\right)$ y da acceso al usuario $\boxed{\text{U}}$ al servicio solicitado si las verificaciones anteriores han sido positivas.
- Calcula $G_{n+1} = E_{n+1} \oplus h(E_{n+1})$ y reemplaza el valor G_n por G_{n+1}

(d) Fase de Cambio de Contraseña Modificada

Esta fase es invocada solamente cuando el usuario desea cambiar su contraseña.

Cuadro 34: Mensajes intercambiados durante la fase cambio de contraseña del protocolo de Vaidya et al modificado.

1. $U \longrightarrow IAS: ID_c, s_1, s_2, E_K(PW, PW_N)$
2. $IAS \longrightarrow U: E_K\left(ID_c, c_{T_N}, u_{N_1}, u_{N_2}\right)$

Paso P1. El usuario U inserta su tarjeta inteligente (SC) en su terminal, o en un lector asociado a su dispositivo personal, e ingresa los valores ID_C , PW y PW_N.

Paso P2. La tarjeta inteligente SC realiza las siguientes operaciones:

- Verifica ID_C. Si ID_C es idéntico al ID_C almacenado por la tarjeta SC, esta continuará con el procedimiento, de lo contrario el proceso de cambio de contraseña se interrumpirá y será rechazada la solicitud actual.
- Obtiene los valores $\begin{cases} K = u_1 \oplus h(PW \oplus h(PW)) \\ h\,(ID_c \oplus x) = u_2 \oplus h(ID_{SC} \parallel PW) \oplus h(K) \end{cases}$
- Calcula $\begin{cases} s_1 = h(ID_c \oplus x) \oplus K \\ s_2 = h(ID_{sc} \parallel K) \oplus h(ID_c \parallel PW) \end{cases}$
- Cifra $E_K(PW, PW_N)$ usando la clave K.

Paso P3. El usuario U envía los valores $\{ID_c, s_1, s_2, E_K(PW, PW_N)\}$ al servidor IAS.

Paso P4. Al recibir $\{ID_c, s_1, s_2, E_K(PW, PW_N)\}$ El servidor IAS procede del siguiente modo:

- Verifica ID_C. Si ID_C es válido, continúa el proceso de autenticación, de lo contrario se interrumpirá y será rechazada la solicitud actual.
- Calcula $K = s_1 \oplus h(ID_c \oplus x)$
- Descifra $E_K(PW, PW_N)$ usando la clave K obtenida
- Calcula $s_2' = h(ID_{sc} \parallel K) \oplus h(ID_c \parallel PW)$
- Compara s_2' con el valor de a_T recibido ($s_2 \overset{?}{=} s_2'$) y si coinciden continúa proceso de autenticación, de lo contrario se interrumpirá y será rechazada la solicitud actual.
- Calcula $\begin{cases} c_{T_N} = h(ID_c \parallel PW_N) \\ u_{N_1} = h\big(PW_N \oplus h(PW)\big) \oplus K \\ u_{N_2} = h(ID_C \oplus x) \oplus h(ID_C \parallel PW_N) \oplus h(K) \end{cases}$
- Cifra $E_K\big(ID_c, c_{T_N}, u_{N_1}, u_{N_2}\big)$

Paso P5. El servidor IAS envía el valor $E_K\big(ID_c, c_{T_N}, u_{N_1}, u_{N_2}\big)$ como respuesta al usuario.

Paso P6. El usuario U procede del siguiente modo:

- Descifra $E_K\big(ID_c, c_{T_N}, u_{N_1}, u_{N_2}\big)$ usando la clave K obtenida inicialmente a partir de la contraseña introducida.
- Calcula $c_{T_N}' = h(ID_c \parallel PW_N)$

- Verifica si c'_{T_N} es el mismo que el valor c_T recibido ($c_{T_N} \overset{?}{=} c'_{T_N}$) y si coinciden continúa proceso de autenticación, de lo contrario se interrumpirá y será rechazada la solicitud actual.
- Reemplaza u_1 , u_2 por los nuevos valores u_{N_1}, u_{N_2} en la tarjeta inteligente.

4.3 Análisis de seguridad del protocolo modificado.

A continuación realizaremos un análisis de la seguridad de nuestro protocolo, en comparación con el de Vaidya et al., basándonos en los ataques viables que un adversario podría intentar contra éste, teniendo en cuenta los elementos usados para la autenticación, la infraestructura de comunicaciones del mismo y la estructura de su diseño. Para ello seremos realistas y no incluiremos ataques extremadamente sofisticados que podrían suponer para nuestro adversario genérico un esfuerzo mucho mayor que el beneficio que este pueda obtener si resulta exitoso. En nuestro análisis los ataques contra la *fase de registro* serán descartados, ya que en la misma el intercambio de información es realizado utilizando un canal seguro de comunicaciones, lo cual es usual en los protocolos de autenticación actuales.

Recordemos además que, tal y como se ha aclarado en el Capítulo 2, en el marco en que estamos trabajando de los protocolos de autenticación de entidades, el análisis de la seguridad y los ataques realizados giran alrededor de los mensajes enviados por los entes activos durante la ejecución del protocolo y la información relacionada con la autenticación los entes principales.

(a) Análisis de los ataques básicos contra nuestro protocolo

En primer lugar descartaremos la posibilidad de algunos ataques básicos cuya definición ya ha sido dada en esta tesis; en estos primeros ataques nuestro adversario tendrá habilidades bastante limitadas y nuestro protocolo resultará igualmente seguro que el de Vaidya et al..

1. Eavesdropping attacks: En nuestro protocolo, al igual que en el de Viaidya et al., todos los secretos importantes para la autenticación son enmascarados mediante operaciones criptográficas, antes de ser transmitidos por canales inseguros de comunicación, de los cuales el atacante podría tener control. De de esta forma, el simple hecho de espiar los canales de comunicación e interceptar los valores transmitidos no le permitirá al atacante obtener información de autenticación útil para romper la seguridad del protocolo.

2. Ataques en sesiones paralelas: Existen dos posibles ataques que hacen uso de sesiones paralelas con el objetivo de intentar vulnerar la seguridad los protocolos de autenticación, los cuales han sido vistos en el Capítulo 2.

- *Reflection attacks*: La falta de simetría en nuestro protocolo, como en el de Vaidya et al., previene la posibilidad de que el mismo tipo de mensajes enviados por uno de sus entes principales, pueda ser usado contra este mismo para realizar una exitosa autenticación. Por otra parte, cada ente principal asume un papel distinto y bien determinado en el protocolo. Tal y como se ha visto en el Capítulo 2, para este tipo de ataque, el adversario interactúa solamente con una de las partes, asumiendo dos papeles distintos en sesiones paralelas; esto sólo puede tener lugar si, además, el ente principal con el que se interactúa puede asumir dos papeles distintos, lo cual no es posible en nuestro protocolo. Dado todo lo anteriormente planteado, este tipo de ataque no se puede realizar contra el mismo.
- *Interleaving attacks*: En este ataque el adversario interactúa con ambas partes, pero utiliza uno de los entes principales como *Oráculo* en una sesión para obtener información de autenticación que pueda ser usada en otra sesión contra el otro ente principal. Sin embargo, ninguna información de autenticación válida en una

sesión dada, lo será en ninguna otra, ya que en nuestro protocolo (al igual que en el de Vaidya et al.) los mensajes de cada sesión dependen del valor actual de HOTP, el cual es diferente para cada sesión. Esto conjugado con la falta de simetría en nuestro protocolo impide que se pueda llevar a cabo este tipo de ataque.

3. Replay attacks: Nuestro protocolo utiliza el mismo mecanismo que el de Vaidya et al. en la *fase de autenticación e inicio de sesión* para evitar este tipo de ataques. En dicha fase, el mecanismo HOTP produce un valor distinto para cada sesión usando un contador ascendente, y este valor es luego incorporado, de distintas formas, a los mensajes de autenticación mediante métodos criptográficos, haciendo los mensajes únicos e irrepetibles. En la *fase de solicitud de acceso al servicio,* esto mismo es garantizado en el protocolo de Vaidya et al. por medio de la *cadena de hash*. En nuestro protocolo el *valor de estado semi-aleatorio* funciona como un parámetro variable que, usado en conjunto con la clave de sesión, permite distinguir mensajes enviados en momentos distintos dentro de una misma sesión, o asociados a sesiones distintas. De este modo, en ambos protocolos, todos los mensajes intercambiados durante la autenticación están unívocamente identificados, y no hay repetición alguna de estos en momentos distintos. Todo lo anterior, sumado a la asimetría en el intercambio de información en todas las fases, evita que sea posible una *replay attack*.

(b) Análisis de otros ataques más sofisticados contra nuestro protocolo

Veamos cómo nuestro protocolo puede también resistir otros tipos de ataques más complicados, algunos de los cuales implican la modificación de mensajes o la combinación de distintas acciones por parte del atacante. Para algunos de estos ataques hemos evidenciado ya, en nuestro análisis de la seguridad del protocolo de Vaidya et al., que este último es inseguro. Para dichos ataques mostraremos cómo en nuestro protocolo estas formas de proceder no son efectivas para el tipo de atacante con el que estamos lidiando en el marco de este trabajo.

4. Password Guessing attacks: Tanto en el protocolo de Vaidya et al. como en el nuestro, las fases que son ejecutadas usando la contraseña de usuario $\boldsymbol{PW}$ como condición necesaria para la autenticación, son la *fase de autenticación e inicio de sesión* y la *fase de cambio de contraseña*. En nuestro caso, el primer mensaje de autenticación de la primera de estas fases (ver Cuadro 32) está formado por el conjunto de valores $\{ID_C\ ,\ s_1\ ,\ s_2\ ,\ G\}$, donde recordemos que $s_1 = h(ID_C \oplus x) \oplus K$, $s_2 = h(ID_C \parallel PW) \oplus h(ID_{SC} \parallel K)$ y $G = h\big(h(ID_C \oplus x) \parallel h(ID_C \parallel PW)\big) \oplus H^i(K, 2C_C)$. Como se puede notar en sus expresiones, estos valores hacen uso de otros secretos combinados criptográficamente con la contraseña de usuario. Dada la forma en que están definidos, un atacante no podrá adivinar la contraseña PW a partir de ninguno de estos valores por sí solos, ya que necesitaría conocer, como mínimo, la clave secreta compartida K. De otra forma, nuestro atacante podría intentar combinar estos valores para tratar de obtener un verificador de la contraseña. Para que esto sea efectivo es necesario que, al menos dos de dichos valores difieran, como mínimo, en un factor (respecto al producto XOR) que sólo dependa de la contraseña PW y un valor público, lo cual no ocurre en este caso. Por otra parte, lo que resta de la información se transmite cifrada usando algún algoritmo de clave simétrica que provee confidencialidad, por lo que los últimos valores enviados son también inservibles para intentar adivinar la contraseña del usuario. En la *fase de cambio de contraseña* ocurre de forma similar, de modo que los únicos valores que se trasmiten sin cifrar son nuevamente $ID_C\ ,\ s_1\ ,\ s_2$, para los cuales nos sirve el mismo análisis anterior. Teniendo en cuenta todo lo analizado, podemos afirmar que un atacante puede obtener la contraseña o un verificador de la misma a partir de la información intercambiada en nuestro protocolo; y por tanto, nuestra propuesta es igualmente resistente a ataques de tipo *password guessing*.

5. Masquerading attacks: Existen básicamente dos tipos de ataques en los que el adversario puede intentar suplantar a uno de los entes principales asumiendo su identidad de forma fraudulenta:

- *Suplantación de la identidad del Usuario*: Para que un atacante logre ser autenticado como *Usuario* legítimo, necesita construir un mensaje válido de autenticación $\{ID_C\ ,\ s_1\ ,\ s_2\ ,\ G\}$ para el servidor IAS. En este caso, aún cuando

los tres primeros valores de este mensaje son estáticos, el valor dinámico G cambia con cada autenticación. Este valor depende de los valores $h(ID_C \oplus x)$, $h(ID_C \| PW)$, K y el contador ascendente C_C y es imprescindible para completar el proceso de autenticación. Dado que ninguno de los valores básicos que componen G puede ser obtenido mediante la combinación de información intercambiada en el protocolo, entonces tampoco será posible construir un valor de G válido que no haya sido usado anteriormente sin el conocimiento de, al menos, la clave secreta "x" del *Servidor* o la contraseña PW del *Usuario*. Esto impide una suplantación de identidad del *Usuario* en la *fase de autenticación y acceso al servicio*.

Por otra parte, un atacante tampoco podrá extraer el valor HOTP actual contando solo con los mensajes interceptados durante ejecuciones del protocolo, y por tanto no podrá así obtener la clave de sesión. Sin ello no le será posible participar en el mecanismo de autenticación bidireccional que se lleva a cabo en la *fase de solicitud de acceso al servicio*.

- *Suplantación de la identidad del Servidor*: Para que un atacante logre ser autenticado como si éste fuese el Servidor *IAS,* tendría que fabricar una respuesta de autenticación *Authresp* válida. Sin embargo, esto no es posible sin el conocimiento de la clave K_i, que depende del valor de HOTP generado en la sesión actual. Como hemos señalado anteriormente, dicho valor de HOTP no puede ser obtenido con sólo interceptar los mensajes intercambiados en el protocolo, por lo que el adversario no podrá suplantar la identidad del Servidor *IAS* contando con información recopilada en modo *eavesdropping*.

Los argumentos expuestos anteriormente, son los mismos que justifican la seguridad del protocolo de Vaidya et al. contra este tipo de ataque, por lo que nuestro protocolo alcanza un nivel de seguridad equivalente ante estas amenazas.

6. Man-in-the-Middle attacks (MiM): En este tipo de ataque el adversario se interpone en las comunicaciones entre dos entes principales, de forma que este modifica los mensajes de ambas partes sin que ninguno de los entes principales que interactúan lo note.

La manipulación de información de autenticación por parte de un atacante para construir mensajes, que al ser verificados por cada parte correspondiente sean dados por válidos, tiene que partir de la manipulación de los secretos sobre la base de los cuales se realizan ambos sentidos de la autenticación en el protocolo. En nuestro protocolo (como en el de Vaidya) se usan tres secretos básicos para la autenticación, los cuales son, como hemos mencionado, la contraseña de usuario PW, la clave secreta del Servidor "x" y la clave secreta compartida K. Estos valores son establecidos y manejados de forma segura durante la *fase de registro del protocolo*. Además, los valores PW y "x", son mezclados, (junto a la clave secreta compartida K) dentro de los valores criptográficos u_1 y u_2 almacenados en la tarjeta inteligente portada por el *Usuario* legítimo. Por otra parte, la *clave secreta compartida* K, que funciona como un secreto común para la autenticación, sólo puede ser obtenida de forma correcta por el *Usuario* mediante su contraseña PW y el valor u_1 almacenado en la tarjeta inteligente, o por el *Servidor* a partir de su clave secreta "x" y el valor s_1 enviado por el *Usuario*. De este modo, K tampoco puede ser manipulada por un adversario sin el conocimiento de la contraseña de usuario o la clave secreta del *Servidor*. Como los procesos de obtención y verificación de los mensajes de autenticación son realizados haciendo uso explícito del valor correcto de la clave secreta compartida K, y esta aparece siempre protegida por operaciones criptográficas seguras, no resulta posible para el atacante modificar un mensaje auténtico para obtener algún otro que pueda ser dado como válido, en el proceso de autenticación, por alguna de las partes activas en el protocolo. Por tanto, nuestro protocolo es resistente a ataques de *man in the middle*, del mismo modo que lo es el de Vaidya et al..

7. Denial of Service (DoS) attacks: En el marco que nos concierne, en este tipo de ataque, el adversario, por medio de acciones ofensivas, trata de impedir que un *Usuario* legítimo pueda participar exitosamente en el protocolo, logrando que el *Servidor* rechace ciertas acciones del mismo, como pueden ser los intentos de inicio de sesión. La forma de lograr esto, para un atacante, supone normalmente que este pueda falsificar el verificador de la contraseña de usuario o invalidar su contraseña sin que este lo note.

Dado que en este caso no se almacena ningún verificador de la contraseña de usuario, este proceder no aplica para nuestro atacante. De otro modo, en nuestro protocolo, la

validación de la contraseña PW se basa siempre en el cumplimiento o no de las igualdades

$$u_1 \oplus h\big(PW \oplus h(PW)\big) = s_1 \oplus h(ID_C \oplus x) = K \qquad (8)$$

Teniendo en cuenta que el valor de la clave secreta compartida K no puede ser cambiada en el protocolo, y que en la *fase de cambio de contraseña* el valor de PW es verificado antes de actualizar los valores en la tarjeta inteligente, el atacante no podrá cambiar ningún valor de verificación dependiente de la contraseña actual. De esta forma, aún cuando el atacante podría importunar al usuario por otras vías, no podrá imitarlo o impedir que este participe exitosamente en el protocolo, siguiendo el procedimiento estipulado en el mismo.

Por otra parte, en nuestro protocolo no es posible importunar al *Usuario* más de lo que sería posible en el protocolo de Viadya et al, por lo que nuestra propuesta es igualmente resistente a este tipo de ataque.

8. Stolen-verifier attacks: Como se ha podido ver, en la mayoría de los casos, el Servidor almacena algún verificador de la contraseña, el cual es mantenido en secreto. Si un atacante logra obtener este verificador de forma ilícita, entonces podrá hacer uso de algún diccionario para intentar adivinar la contraseña.

En nuestro protocolo (al igual que en el de Vaidya et al.) no se almacena ningún tipo de verificador de contraseña en el Servidor por lo que este tipo de ataque es imposible.

9. Smart Card Loss attack: Cuando la tarjeta inteligente de un usuario cae en manos de un atacante, este puede intentar usarla para acceder al sistema suplantando al usuario legítimo o cambiar la contraseña del mismo. De otro modo, podría intentar usarla para tratar de adivinar la contraseña actual del usuario, probando valores con ayuda de un *Diccionario* suficientemente bueno. Incluso, con los recursos adecuados, este podría extraer los valores almacenados en la tarjeta inteligente y usarlos para obtener un verificador de la contraseña o construir un mansaje válido de autenticación.

En primer lugar, un atacante en posesión de la tarjeta inteligente no puede usarla por si sola para autenticarse con el *Sistema* si este no conoce la contraseña del usuario ya que la autenticación se lleva a cabo en base a estos dos factores; y tampoco podrá intentar adivinar la contraseña puesto que la tarjeta cuenta con un valor C_M que limita el número

máximo de intentos fallidos con una contraseña dada, luego de lo cual el usuario deberá volver a registrarse con el *Sistema*. Por otra parte, si al atacante logra extraer los valores almacenados en la tarjeta inteligente e interceptar los mensajes trasmitidos durante ejecuciones del protocolo, lo más que podrá acumular son los valores criptográficos:

- $u_1 = h\big(PW \oplus h(PW)\big) \oplus K$
  $u_2 = h(ID_C \oplus x) \oplus h(ID_C \parallel PW) \oplus h(K)$
- $s_1 = h(ID_C \oplus x) \oplus K$
  $s_2 = h(ID_C \parallel PW) \oplus h(ID_{SC} \parallel K)$
- $G = h\big(h(ID_C \oplus x) \parallel h(ID_C \parallel PW)\big) \oplus H^i(K, 2C_C)$
- $E_{K_{IAS-HG}}\big(ID_C, ID_{IAS}, S_{k_i}, E_1, N, T_{exp}\big)$
- $E_{K_i}(ID_C, ID_{IAS}, TKG_C, E_1, S)$

Los dos últimos valores se encuentran totalmente cifrados con claves distintas usando algoritmos seguros, por lo que solo se podría considerar un posible ataque contra las claves usadas. De estas, la clave pre-compartida K_{IAS-HG} también está fuera de nuestro análisis pues forma parte de la infraestructura pre-existente para el desarrollo del protocolo. Si se realizan las combinaciones posibles de los primeros cinco valores mediante operaciones XOR se podrá observar que no es posible obtener ningún valor que solo dependa de la contraseña PW y algún valor público como pueden ser los identificadores; como mismo tampoco es posible obtener de este modo los valores $h(ID_C \oplus x)$ o K. Ello implica que el adversario no podrá obtener la clave K_i o un verificador de la contraseña y, más aún, tampoco podrá fabricar un mensaje de autenticación válido, ya que todos estos mensajes dependen de los valores mencionados de una u otra forma. Dado todo lo anterior, se puede concluir que nuestro protocolo bloquea la posibilidad de efectuar un ataque efectivo de *verificación de contraseñas con pérdida de la tarjeta inteligente*, como el que se ha planteado contra el protocolo de Vaidya et al., o siguiendo alguna otra táctica previsible.

10. Secret key Forward Secrecy: Esta propiedad se cumple si, aún cuando el adversario ha podido obtener la clave secreta "x" del *Servidor*, este no puede obtener las calves de sesión generadas hasta el momento en el protocolo.

Un atacante en posesión de a clave secreta "x" puede calcular el valor $h(ID_C \oplus x)$. A partir de esto y los mensajes interceptados puede entonces obtener el los valores correctos de K y $h(ID_C \| PW)$. Con ello podrá calcular incluso la máscara $h\big(h(ID_C \oplus x) \| h(ID_C \| PW)\big)$ y a partir de G obtener el valor $H^i(K, 2C_C)$ actual. Sin embargo, nuestro adversario no podrá obtener, procediendo de este modo, otros valores generados por el algoritmo HOTP, más que aquellos valores generados usando valores pares $2C_C$ como factor de cambio. De este modo, no se verán comprometidas las claves $K_i = H^i(K, 2C_S + 1)$ que son usadas para obtener las claves de sesión $S_{k_i} = h(K_i \oplus S)$, ya que K_i es obtenido usando valores impares $2C_S + 1$ como factor de cambio.

Ahora notemos que en el protocolo de Vaidya. et al., las claves de sesión son generadas usando la misma secuencia de los valores de cambio C_S con la que se obtiene el valor HOTP; mientras que en nuestro protocolo hemos utilizado dos sub-secuencias no coincidentes para separar los valores del factor de cambio que alimentan a los HOTPs usados dentro de G, de los usados en las claves K_i. Por tal razón nuestra propuesta cumple en mejor medida la propiedad de *secret-key forward secrecy*.

(c) Seguridad brindada por el mecanismo de los Valores de Estados Semi-Aleatorios

Para la mejor comprensión de la forma en que hemos usado el *valor de estados semi-aleatorios* para remplazar la *cadena de hash* del protocolo de Vaidya, realizaremos un análisis detallado de su aplicación en la *fase de solicitud de acceso al servicio*, incluyendo observaciones comparativas con respecto al uso de la *cadena de hash* dentro de esta misma fase en el protocolo de Vaidya.

Para comenzar señalemos que ambos mecanismos dependen esencialmente de dos secretos básicos, a partir de los cuales es generada la información de autenticación. Como en todo mecanismo de autenticación basado en secretos, es necesario asumir que el conjunto total de valores secretos en los cuales reside la seguridad del mecanismo no será comprometido. Sin embargo, es importante garantizar que el conocimiento parcial de los

secretos básicos no comprometa a los valores restantes durante la ejecución del mecanismo de autenticación.

En el caso del protocolo de Vaidya, el conjunto de secretos básicos para la *fase de solicitud de acceso al servicio* está formado por la semilla aleatoria S y la clave de sesión S_K, que constituyen valores estáticos dentro de cada ejecución de dicha fase.

A partir de estos valores es obtenida completamente la *cadena de hash* y se calcula el valor $R = h(S \oplus C \oplus S_K)$, dentro del cual el contador C es enviado en claro por el punto de acceso HG.

Dado que siempre son necesarios estos dos valores para obtener los mensajes de autenticación y dichos valores son independientes, la seguridad del mecanismo no se verá afectada si solo uno de ellos es comprometido.

En el caso de nuestra propuesta el conjunto de secretos básicos está formado por la clave de sesión S_{k_i} y el *valor de estado* actual de una de las partes, ya sea E_n mantenido por el *Usuario* o G_n mantenido por el Punto de Acceso HG. En nuestro caso el valor de estados cambia con cada acceso al servicio, aportando así dinamismo al conjunto de secretos básicos. Estos valores son además independientes entre sí, por lo que ninguno de ellos puede ser obtenido a partir del otro por sí solo.

Mostremos que en ningún momento de la ejecución de la *fase de solicitud de acceso al servicio* el comprometimiento de sólo uno de estos valores compromete el conjunto básico de valores secretos.

En el *n-esimo* acceso al servicio el U mantendrá los secretos $\{S_{k_i}, E_n\}$ mientras que HG mantendrá $\{S_{k_i}, G_n\}$ donde $G_n = E_n \oplus h(E_n)$. Observemos que G_n funciona como un valor trampa de E_n, de forma que no es posible obtener E_n a partir de G_n por si solo.

El intercambio de información durante el primer acceso al servicio será como sigue:

Cuadro 35: Intercambio de mensajes durante la fase solicitud de acceso al servicio del protocolo de Vaidya et al. modificado.

1. $HG \longrightarrow U : A_n \oplus h(S_{k_i}),\ h(A_n \parallel G_n)$
2. $U \longrightarrow HG : h(E_n) \oplus h(A_n)$

Considerando que todos los mensajes de autenticación transmitidos pueden ser capturados por un atacante en modo *eavesdroping*, la estructura de los mismos brinda un adecuado nivel de seguridad, de modo que:

- Aún si la cave de sesión fuese revelada, ninguno de los dos sentidos de la autenticación puede ser llevada a cabo, ya que el primer sentido de la misa se basa fundamentalmente en el conocimiento del valor de estado de HG, como condición necesaria para poder generar el valor acertado de $h(A_n \parallel G_n)$. Sin ello tampoco U emitirá posteriormente el valor $E_n \oplus h(A_n)$.

  Por otra parte, aunque usando el valor $h(S_{k_i})$ y a partir de los mensajes de autenticación de accesos a servicios anteriores se pueden obtener a lo sumo las sucesiones de valores anteriores $A_1, A_2, \ldots, A_{n-1}$ y $h(E_1), h(E_2), \ldots, h(E_{n-1})$, que no permiten calcular el valor de estado actual

  $$E_n = h\left(\left(E_{n-1} \oplus h(A_{n-1})\right) \parallel S_{k_i}\right).$$

- Si el valor de estado G_n de HG se viera comprometido no sería posible obtener la clave de sesión, y sin ello tampoco los valores A_n y E_n, ya que G_n solo se usa dentro del *valor hash* $h(A_n \parallel G_n)$.
- Además, último si el valor de estado E_n se ve comprometido, esto solo permitiría calcular G_n y lo cual no es suficiente para generar un mensaje válido de autenticación $A_n \oplus h(S_{k_i})$, $h(A_n \parallel G_n)$ sin el conocimiento del valor $h(S_{k_i})$; ni es posible obtener este valor después haber observado una ejecución exitosa de la autenticación durante el acceso al servicio usando el estado E_n, ya que a lo sumo podrá obtener $h(A_n)$, que no puede ser usado con el valor $A_n \oplus h(S_{k_i})$ para obtener $h(S_{k_i})$.
- Por último, dado lo anteriormente analizado y el hecho de que siempre el próximo valor de estado E_{n+1} depende de los tres valores actuales E_n, A_n y S_{k_i} , no será posible calcular E_{n+1} con el conocimiento parcial del conjunto de secretos básicos.

Para concluir este acápite señalemos que la decisión de usar un valor de estado distinto para U y HG se debe a que nuestra intención es preservar las buenas características del mecanismo de la *cadena de hash* utilizada en el protocolo de Vaidya., en el cual no es posible obtener el valor *hash* P_i que será utilizado por U a partir del valor P_{i-1} usado previamente por HG.

De forma similar, en nuestro caso tampoco será posible obtener el valor de estado E_n de U a partir del valor de estado G_n de HG que es usado siempre con anterioridad, simulando sí esta característica de las *cadenas de hash*.

4.4 Comparación con el protocolo de Vaidya et al.

Dado que las modificaciones realizadas al protocolo de Vaidya et al. están sólo relacionadas con los mensajes que se intercambian y las acciones criptográficas que realizan sus entes principales para obtener y verificar dichos mensajes, los cambios realizados no han afectado la estructura o los principios de funcionamiento del protocolo; por lo que nuestro protocolo ha heredado las buenas características y funcionalidades del protocolo de Vaidya. De este modo, nuestra propuesta cumple en la misma medida los *aspectos para la conformación de un buen protocolo de autenticación* analizados en el acápite 2.1(a), y por encima de ello, mejora de forma notable los aspectos 5 y 6 relacionados con la eficiencia del protocolo.

En nuestro protocolo, durante la *fase de registro*, sólo se almacenan dos valores criptográficos, u_1 y u_2, en la tarjeta inteligente para la obtención de información de autenticación por parte del Usuario, en vez de los tres valores v_T, g_T y k_T necesarios en el protocolo de Vaidya et al. Además, con los cambios realizados a la *fase de autenticación e inicio de sesión*, se reduce ligeramente el total de operaciones necesarias. Así, puede verificarse para dicha fase, que en nuestro protocolo sólo se necesitan computar 14 operaciones *hash* y 9 operaciones XOR, en contraste con las 15 operaciones *hash* y 14 operaciones XOR que se realizan en el protocolo de Vaidya et al. según cálculos de los propios autores (Vaidya et al., 2011). En esta misma fase también se ha

reducido la cantidad de información a cifrar con criptografía de clave simétrica y la longitud de los mensajes $AuthResp$ y TKG. Esto se debe a que tanto $AuthResp$ como en TKG se obtienen cifrando dos valores menos. Al mismo tiempo se ha logrado proporcionar integridad al tiquete de autenticación TKG, lo cual no se brindaba en el protocolo de Vaidya et al.

La *fase de solicitud de acceso al servicio* ha sido modificada por completo en nuestra propuesta. En la misma hemos reemplazado el costoso mecanismo basado en la *cadena de hash* por una sincronización secuencial combinado con desafío-respuesta mediante números aleatorios, cuyos valores resultantes hemos llamado en esta tesis *estados semi-aleatorios*. Este nuevo mecanismo introducido, no sacrifica la seguridad ya proporcionada por el protocolo de Vaidya et al., sin embargo resulta en una mejora bastante notable de la eficiencia computacional de esta fase.

En el artículo (Vaidya et al. 2011) donde los autores proponen su esquema, estos sólo realizan comparaciones de la eficiencia computacional de su protocolo, basado en las operaciones criptográficas del mismo, durante la ejecución de las fases de *registro* (*R*) y de *autenticación e inicio de sesión* (*LA*), sin considerar la *fase de solicitud de acceso al servicio* (*SR*). Sin embargo, esta es la fase más frecuente del protocolo. A pesar de que la cantidad de operaciones criptográficas del protocolo propuesto por Vaidya et al. varía con cada acceso al servicio a causa de la sincronización basada en la *cadena de hash* usada, las operaciones realizadas durante la *fase de solicitud de acceso al servicio* pueden ser calculadas en función del valor N que representa el número máximo de accesos al servicio permitido dentro de una sesión. Este número es usado como exponente máximo de la *cadena de hash*. De este modo, son necesarias N operaciones hash para inicializar dicha cadena, a partir de lo cual el número de operaciones va descendiendo linealmente. En esta progresión aritmética podemos saber la cantidad de operaciones hash y XOR que se realizan durante el *n-ésimo* acceso al servicio a lo largo de una ejecución exitosa de la fase correspondiente. Esto nos permite comparar la cantidad de operaciones necesarias en nuestro protocolo, con las necesarias en el protocolo de Vaidya et al., durante la ejecución de la *fase de solicitud de acceso al servicio*. Teniendo en cuenta que es necesario realizar una distinción entre la primera solicitud de acceso al servicio y las demás ejecuciones de

esta fase, ya que es preciso inicializar ciertos valores primeramente, hemos obtenido la siguiente tabla a modo de comparación.

Cuadro 36: Comparación entre el protocolo de Vaidya et al. y el modificado según la cantidad de operaciones Hash y XOR en la fase de solicitud de acceso a servicios.

Ente principal	*Protocolo Modificado*	*Protocolo de Vaidya et al.*
$\boxed{HG}$	$1ra\ vez$: $\mathbf{7h, 6} \oplus$ $n-$ésima vez: $\mathbf{5h, 5} \oplus$	$1ra\ vez$: $\mathbf{(N+4)h, 5} \oplus$ $n-$ésima vez: $\mathbf{2h, 4} \oplus$
$\boxed{U}$	$1ra\ vez$: $\mathbf{5h, 4} \oplus$ $n-$ésima vez: $\mathbf{4h, 4} \oplus$	$1ra\ vez$: $\mathbf{(2N)h, 4} \oplus$ $n-$ésima vez: $\mathbf{(N-n+1)h, 3} \oplus$

Nota: h = *operación hash*, $\oplus =$ $operación\ XOR$

En dicha tabla se puede observar cómo en nuestra prouesta la mayor parte de las operaciones *hash* de esta fase son calculadas por el punto de acceso HG, a diferencia de lo que ocurre en el esquema de Vaidya et al., en el cual el *Usuario* carga con la mayoría de estas operaciones al tener que generar los valores de la *cadena de hash*. Las modificaciones que realizamos en esta fase están orientadas disminuir, tanto como sea posible, el consumo energético del *Usuario*, mediante la reducción de las operaciones que éste necesita computar para completar su acceso a los servicios de la red, ya que esta es una preocupación general teninendo en cuenta que hoy día gran cantidad de dispositivos inalambricos participan en estos entornos. Este objetivo es logrado mediante una drástica reducción en la cantidad de operaciones *hash* que este debe realizar en la *fase de solicitud de acceso al servicio*, lo que conduce a elevar la eficiencia computacional del protocolo presentado en general.

Para que se vea más claramente la diferencia entre la cantidad de operaciones *hash* entre el protocolo propuesto y el protocolo de Vaidya et al., para cada acceso al servicio dentro de una sesión dada, hemos graficado el seguimiento de estas cantidades para un valor de $N = 100$, que es el mismo valor de este parámetro usado como ejemplo en (Vaidya et al., 2011).

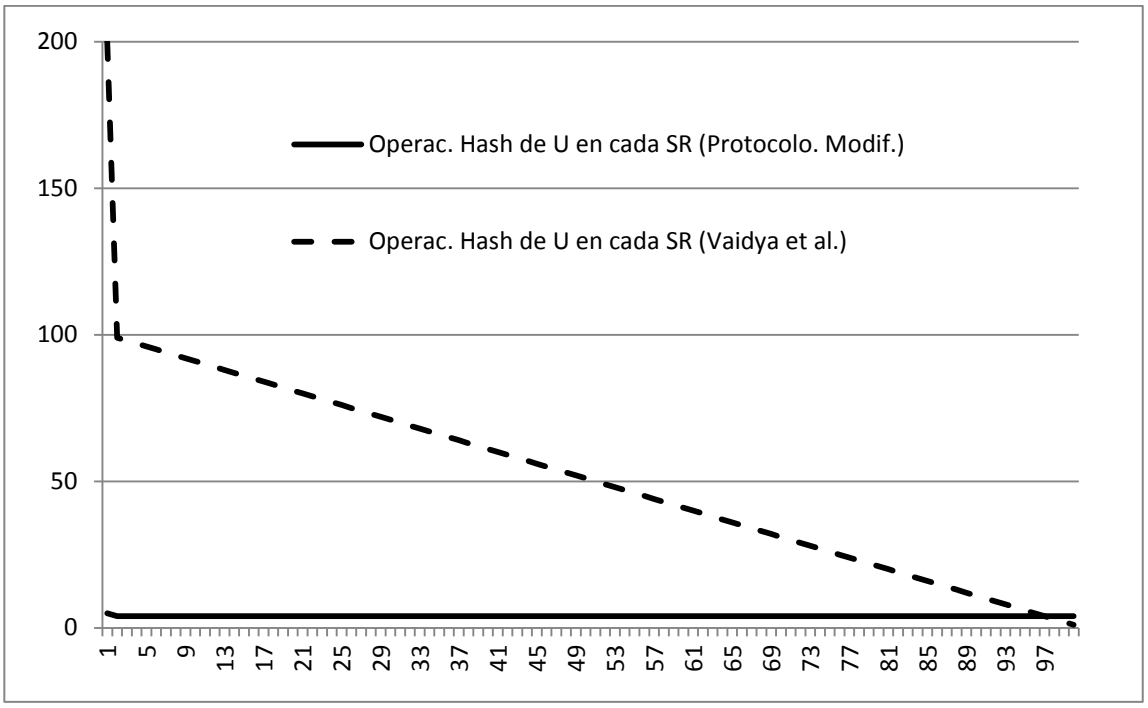


Figura 3: Gráfico comparativo de la cantidad de operaciones hash computadas por el Usuario en el Protocolo Modificado y el protocolo propuesto por Vaidya et al (2011) a lo largo de una sesión, para un valor de N=100.

En la Fig. 3 se puede apreciar cómo a pesar de que la cantidad de operaciones *hash* que debe computar el *Usuario* en cada ejecución de la *fase de acceso al servicio* del protocolo de Vaidya et al. va disminuyendo, en comparación con la vez anterior, esta cantidad de operaciones necesarias se mantiene muy por encima de las necesarias en nuestra propuesta la gran mayoría del tiempo. Como consecuencia, al cabo de *N* ejecuciones exitosas de la *fase de acceso al servicio*, la diferencia entre la cantidad de operaciones *hash* realizadas hasta ese momento dentro de la misma sesión se hará aún más notable. Para ilustrar este hecho hemos agregado una tabla basada en la cantidad de operaciones *hash* computadas hasta ese momento en cada *acceso al servicio* y hemos agregado la gráfica de la diferencia acumulativa.

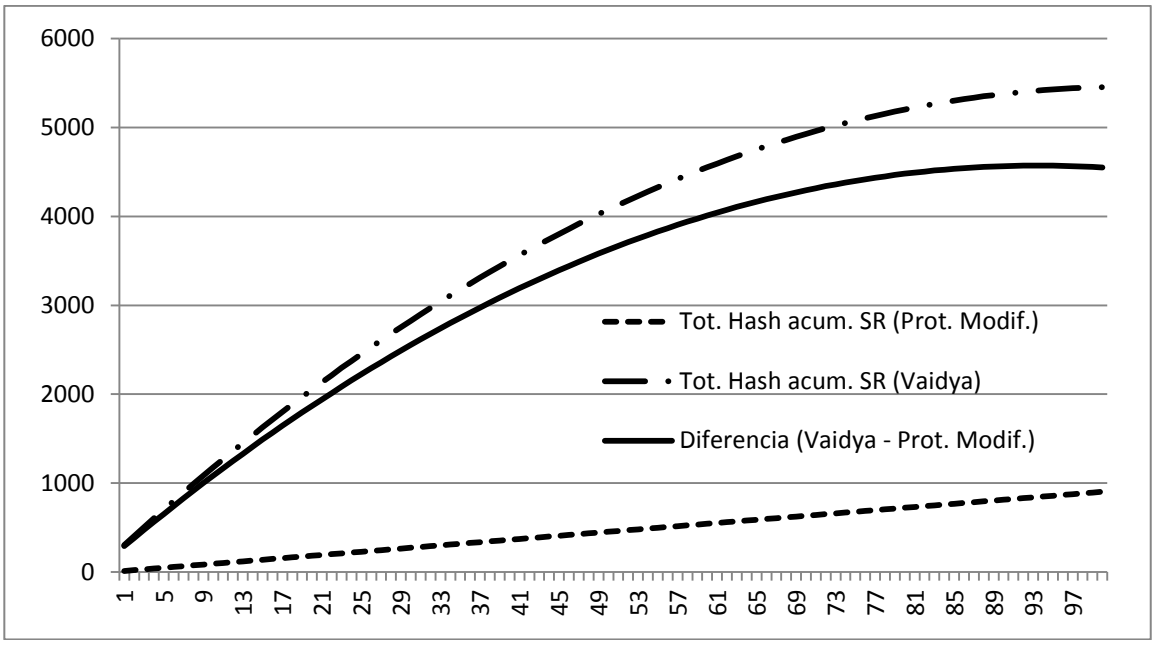


Figura 4: Gráfico comparativo de la cantidad total acumulativa de operaciones hash computadas en el Protocolo Modificado y el protocolo propuesto por Vaidya et al. (2011) a lo largo de una sesión, para un valor de N=100.

Nótese que aún cuando en nuestra propuesta el punto de acceso HG debe realizar algunas operaciones *hash* más que en el protocolo de Vaidya et al., de forma general, después unos cuantos accesos al servicio dentro de una misma sesión, la cantidad de operaciones *hash* computadas hasta ese momento en el protocolo de Vaidya et al., es mucho mayor que la cantidad de operaciones *hash* necesarias en el protocolo modificado. En particular, para un valor de $N = 100$, al término de la sesión se habrán realizado en nuestro protocolo 4549 operaciones *hash* menos que en el protocolo de Vaidya et al. que hemos modificado en este trabajo.

Para terminar de ilustrar en qué medida es más eficiente computacionalmente nuestro protocolo que el protocolo de Vaidya et al. mostramos a continuación otra tabla similar a la de la Fig. 4 pero basada en la cantidad total de operaciones, o sea, incluyendo esta vez las operaciones XOR dentro de nuestros cálculos.

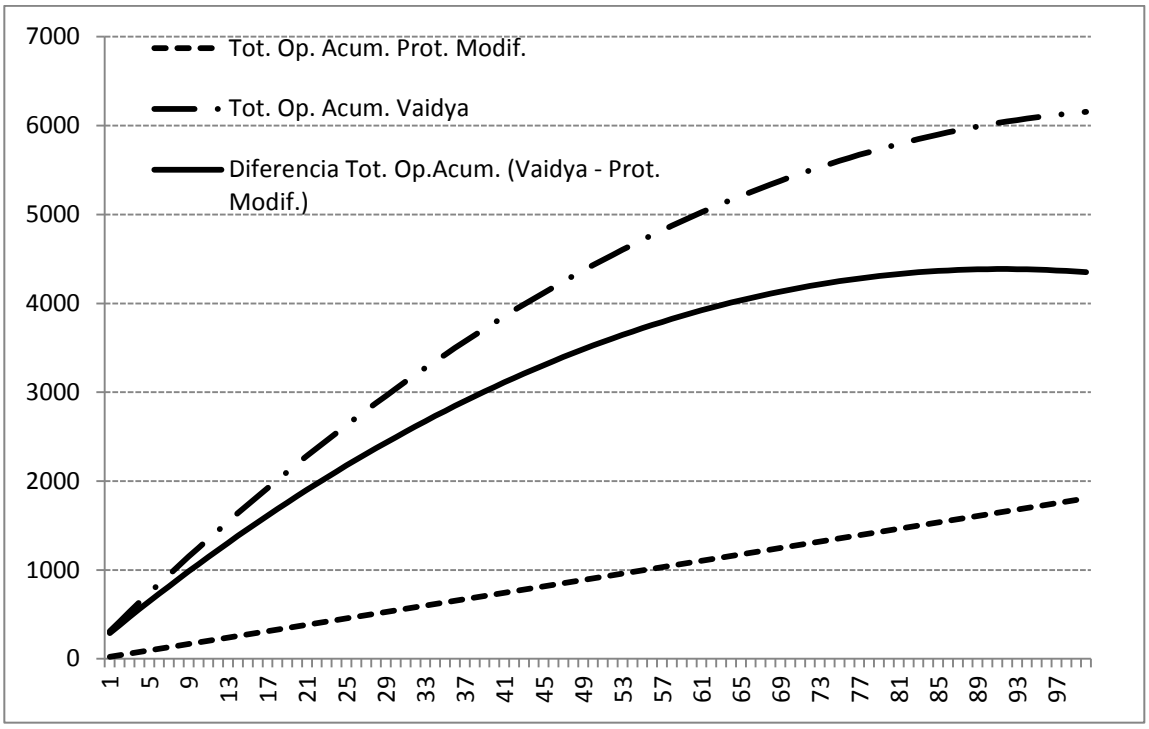


Figura 5: Gráfico comparativo de la cantidad total acumulativa de operaciones (hash y XOR) computadas en el Protocolo Modificado y el protocolo propuesto por Vaidya et al. (2011) a lo largo de una sesión, para un valor de N=100.

En la Fig. 5 se puede apreciar cómo, a pesar de que las modificaciones que hemos realizado a la *fase de solicitud de acceso al servicio*, no han estado en función de disminuir las operaciones XOR, la drástica reducción que se ha logrado en la cantidad de operaciones hash necesarias conduce a una notable reducción de la cantidad total de operaciones, considerando ambos tipos (ver anexos 1, 2 y 3). Dado que estas dos son las operaciones más utilizadas en ambos protocolos analizados, de aquí se deduce que las modificaciones realizadas conllevan a un notable aumento en la eficiencia computacional. No obstante, podemos agregar que, aunque a lo largo de todo nuestro protocolo se realiza la misma cantidad de operaciones de cifrado y descifrado que en el protocolo de Vaidya et al.; en nuestro caso, cada vez que se realizan estas operaciones se cifra menos información, por lo que también se logra una reducción del costo computacional en estos pasos.

Este último hecho planteado también conduce a una reducción de la información que se intercambia en la última parte de la *fase de autenticación e inicio de sesión*, ya que en

nuestra propuesta se envían los valores $E_{K_i}(ID_C, ID_{IAS}, TKG_C, E_1, S)$ y $E_{K_{IAS-HG}}(ID_C, ID_{IAS}, S_{k_i}, E_1, T_{exp})$, que resultan más cortos que los valores $E_{K_i}(ID_C, ID_{IAS}, N_a, A_S, N, S)$ y $E_{K_{IAS-HG}}(ID_C, ID_{IAS}, N_a, K_i, N, S, T_{exp})$ enviados en esta fase en el protocolo de Vaidya et al. Por otra parte, en la *fase de solicitud de acceso al servicio* se envía un valor menos en nuestro caso. Teniendo en cuenta que además ningún mensaje intercambiado en el resto del protocolo tiene (por cuestiones de diseño) una longitud mayor que su mensaje equivalente del protocolo que se ha modificado, se puede decir que en nuestra propuesta se mejora ligeramente la eficiencia del protocolo en las comunicaciones.

Teniendo en cuenta el artículo de la propuesta Kim et al. (2011), donde se realizan modificaciones al protocolo de Vaidya et al., y las cuestiones ya mencionadas sobre este trabajo, podemos resumir, en la siguiente tabla, una comparación de la eficiencia entre estos tres protocolos mencionados usando como base el protocolo de Vaidya.

Cuadro 37: Comparación según eficiencia computacional y de las comunicaciones entre los protocolos de Kim et al. y Vaidya modificado.

Protocolos	Autenticación - Inicio de Sesión	Solicitud de Acceso a Servicios
Kim et al. (2011)	$<$ *Eficiencia Computacional*	$=$ *Eficiencia Computacional*
	$=$ *Eficiencia en las Comunicaciones*	$=$ *Eficiencia en las Comunicaciones*
Vidya modificado (nuestra propuesta)	$>$ *Eficiencia Computacional*	$>>$ *Eficiencia Computacional*
	$>$ *Eficiencia en las Comunicaciones*	$>$ *Eficiencia en las Comunicaciones*

Por otra parte, en nuestro protocolo se ha logrado corregir la vulnerabilidad del protocolo de Vaidya. et al (2011) contra ataques con robo de la tarjeta inteligente, incrementado el nivel de seguridad del mismo. Teniendo en cuenta nuestro análisis de seguridad realizado en el epígrafe 3.4 de este trabajo y los análisis de seguridad realizados por Vaidya et al (2011) y Kim et al (2011) para sus propuestas, se puede realizar una comparación entre nuestro esquema y estos otros dos, en base a los ataques más significativos, lo cual se aprecia en la siguiente tabla.

Cuadro 38: Comparación del protocolo propuesto con el de Vaidya et al., en base a los ataques analizados.

Ataques enumerados en 3.3 (a) y (b)	1	2	3	4	5	6	7	8	9	10
Seguridad de nuestra propuesta (comparado con el protocolo de Vaidya)	=	=	=	=	=	=	=	=	>	>

Nota: = (igualmente seguro), > (más seguro)

CONCLUSIONES

En este trabajo se ha hecho una revisión y un análisis, lo más exhaustivo posible, del protocolo de autenticación robusta propuesto recientemente por de Vaidya et al. (2011). Para ello, primeramente nos hemos dado a la tarea de entender, y mostrar, los principios de funcionamiento de los protocolos de autenticación de entidades, así como algunos aspectos básicos y fundamentales para el buen diseño y la evaluación de los mismos. Posteriormente nos hemos adentrado en el estudio de la autenticación basada en secretos compartidos, donde hemos identificado las posibles amenazas básicas contra los mecanismos usados para ello, y las distintas técnicas para brindar fortaleza a este proceso. Como consecuencia del surgimiento de amenazas cada vez menos simples, se llega a la necesidad de las contraseñas de un solo uso (OTPs) y la autenticación de doble factor. Un recorrido ascendente por los protocolos de autenticación robusta, hasta llegar al protocolo de Vaidya et al., nos ha servido para observar las distintas formas propuestas de implementación de los mecanismos estudiados y las diferentes combinaciones entre los mismos, con el objetivo de incrementar cada vez más la seguridad en los protocolos de autenticación. A la vez, este estudio nos ha permitido notar las fallas que han presentado las diferentes implementaciones de los mecanismos de autenticación mencionados y las distintas medidas que se han ido proponiendo para su corrección. De esta forma, en nuestro trabajo se evidencia la evolución de los protocolos de autenticación robusta, lo cual ayuda a la comprensión de la complejidad que estos presentan actualmente.

Por otra parte, el reto de garantizar niveles adecuados de seguridad resulta aún mayor cuando se quieren proponer protocolos ligeros, basados sobre todo en operaciones criptográficas como *Hash* y *XOR*, y en los cuales se evita tanto como sea posible el uso de los métodos clásicos de cifrado. Para este tipo de mecanismos de autenticación, basados en primitivas criptográficas ligeras, existe escasa bibliografía disponible, ya que en su gran mayoría la bibliografía relacionada con la autenticación se basa en criptografía de clave simétrica o de clave privada, y está dirigida en gran parte al intercambio seguro

de claves. Por esta razón, tenemos la intención de que nuestro trabajo sirva como un buen material de consulta para aquellos que quieran profundizar en los protocolos de autenticación robusta basados en secretos compartidos y primitivas criptográficas ligeras, como *Hash* y *XOR*.

El estudio abarcado nos ha servido como base indispensable para la compresión del protocolo de Vaidya et al.; lo cual nos ha permitido realizar un análisis integral de dicho protocolo, haciendo énfasis en los aspectos de seguridad y rendimiento computacional. Mediante el análisis realizado pudimos identificar las siguientes deficiencias del protocolo de Vaidya:

1. Es vulnerable a *ataques de verificación de contraseña con pérdida de la tarjeta inteligente*, como se ha mostrado con el ataque implementado por Kim et al. (ver epígrafe 4.1 c)).
2. No cumple la propiedad *secret-key forward secrecy*, lo cual hemos evidenciado mediante el segundo ataque que hemos propuesto, el cual compromete los valores HOTP usados para la generación de claves de sesión.
3. No brinda integridad al tiquete de autenticación TKG entregado al *Usuario* durante la última parte del intercambio de información de la fase de *autenticación e inicio de sesión*.
4. Usa un mecanismo relativamente costoso para la autenticación, desde el punto de vista computacional, durante la *fase de solicitud de acceso al servicio*, ya que se basa en una *cadena de hash* y en se le encarga al *Usuario* una gran parte de las operaciones realizadas.

Teniendo muy en cuenta los aprendizajes de todo el estudio realizado, hemos propuesto modificaciones para mejorar el protocolo de Vaidya et al., las cuales han dado lugar a un nuevo protocolo, dentro del cual hemos respetado los principios fundamentales de funcionamiento y las buenas propiedades del protocolo de Vidya et al.. Este protocolo propuesto ha sido desglosado, analizado y comparado con el de Vaidya et al., dejando ver cómo en el mismo se da solución a las deficiencias anteriormente enumeradas de dicho protocolo, de modo que:

1. Se ha eliminado la dependencia entre los valores del protocolo de Viadya et al. que abrían la posibilidad de ataques previsibles de *verificación de contraseña con pérdida de la tarjeta inteligente*.
2. Se han separado los valores HOTP que se usan en el protocolo de Vaidya et al para la autenticación en la segunda fase del protocolo, de aquellos que son utilizados para la obtención de las claves de sesión; con lo cual el ataque que hemos planteado contra dicho protocolo no es efectivo, y se hace más difícil implementar cualquier otro ataque que viole la propiedad de *secret-key forward secrecy*.
3. Se ha proporcionado integridad al tiquete de autenticación TKG entregado al *Usuario* durante la última parte del intercambio de información de la fase de *autenticación e inicio de sesión*, sin incremento de la información intercambiada durante esta fase.
4. Se ha mejorado notablemente la eficiencia computacional del protocolo utilizando un mecanismo de sincronización secuencial combinado con desafío respuesta, que hemos usado para reemplazar la *cadena de hash* de la *fase de solicitud de acceso al servicio*. Dicho mecanismo no ha sido encontrado en la bibliografía revisada, y a la secuencia de valores del mismo le hemos llamado *valores de estado semi-aleatorios*. Esto ha conducido a una drástica reducción de la cantidad de operaciones que necesita realizar el *Usuario* para completar el proceso de autenticación necesario para acceder a los servicios de la red.

Cabe señalar que también hemos hecho otras modificaciones menos significativas al protocolo de Vaidya et al., con las cuales se ha reducido ligeramente el tamaño de algunos mensajes intercambiados y el almacenamiento de información en las dos primeras fases.

Finalmente hacemos notar que, aunque Kim et al. (2011) propusieron modificaciones al protocolo de Vaidya et al. para mejorar la seguridad proporcionada por este, su propuesta es más costosa computacionalmente. A pesar de ello, hemos mostrado mediante el ataque realizado en el epígrafe 4.1 f) que este protocolo tampoco es tan seguro como sus autores han planteado. Nuestro protocolo logra un nivel de seguridad

mayor que el de Vaidya et al., pero resulta bastante más eficiente, por lo que nos parece una alternativa que podría ser preferible al de Kim et al. (2011) en aquellos escenarios en los cuales la mayor parte de los dispositivos de *Usuario* son inalámbricos, por lo que necesitan ser conservadores en el consumo de potencia.

Con todos estos aspectos hemos alcanzado los objetivos específicos de nuestro trabajo; de modo que finalmente hemos logrado modificar el protocolo de Vaidya et al. (2011) para obtener un protocolo más seguro y eficiente, cumpliendo así con el objetivo de esta Tesis.

RECOMENDACIONES Y TRABAJOS FUTUROS

Aunque nuestro protocolo ha sido planteado para el escenario de las redes digitales en el hogar, este puede ser fácilmente aplicado a otros entornos en los que se use el mismo modelo de infraestructura. Por otra parte, en el escenario de las transacciones monetarias que se basan en tarjetas bancarias se ha hecho cada vez más importante la protección de los clientes contra los posibles ataques que se pueden implementar con el robo de tarjetas. Sería interesante tratar de adaptar a este tipo de escenario el esquema y los mecanismos usados en los últimos protocolos con los que hemos tratado en esta tesis, incluyendo nuestra propuesta; ya que el mecanismo de autenticación usando un número PIN y una tarjeta bancaria es un caso particular del tipo de autenticación de doble factor, en el cual se basan los protocolos principales de nuestro trabajo.

El mecanismo mixto de los *Valores de Estado Semi-Aleatorios*, que ha contribuido a elevar notablemente la eficiencia computacional del protocolo de Vaidya modificado, posee características interesantes que permiten realizar autenticación bidireccional con bajo costo computacional en comparación con otros mecanismos. Una evaluación más rigurosa y completa de este mecanismo en sí, en comparación con el resto de los mecanismos de OTPs existentes, sería un buen paso para revelar las conveniencias posibles aplicaciones que puede tener dicho mecanismo usado en este trabajo, como un aporte propio de nuestra investigación.

Aunque se han dado argumentos razonables para justificar las afirmaciones planteadas con respecto a la seguridad del protocolo propuesto en esta tesis, y se ha seguido el estilo generalmente usado en la gran mayoría de los artículos revisados para la conformación del estado del arte de nuestro trabajo, sería muy bueno realizar un análisis del protocolo propuesto por medio de métodos más formales. Para ello podría considerarse la lógica de protocolos no monotónicos, conocida como lógica de Borrows; sin embargo, un tal

análisis formal deber llevarse a cabo con bastante cuidado, siendo lo más exhaustivo posible.

Teniendo en cuenta lo anterior, y dado que todo protocolo propuesto debe ser sometido a una revisión seria antes de ponerse en práctica en algún escenario, nuestras próximas líneas de trabajo estarán enfocadas a formalizar al mayor nivel posible todos los análisis realizados, y a tratar de adicionar nuevas modificaciones que ayuden a resolver vulnerabilidades aún no resueltas por los protocolos actuales del mismo tipo.

BIBLIOGRAFÍA

Bird, R., Copal, I., Herzberg, A., Janson, P. A., Kutten, S., & Yung, M. (1993). Systematic design of a family of attack-resistant authentication protocols. *IEEE Journal on Selected Areas in Communications*, *11*(5), 679–693.

Burrows, M., Abadi, M., & Needham, R. (1990). A Logic of Authentication. *ACM Transactions on Computer Systems*, *8*(1), 18–36.

Chan, M., Estève, D., Escriba, C., & Campo, E. (2008). A review of smart homes-present state and future challenges. *Computer methods and programs in biomedicine*, *91*(1), 55–81. doi:10.1016/j.cmpb.2008.02.001

Chang, Y.-F., Chang, C.-C., & Kuo, J.-Y. (2004). A secure one-time password authentication scheme using smart cards without limiting login times. *ACM SIGOPS Operating Systems Review*, *38*(4), 80–90.

Chen, C.-M., & Ku, W.-C. (2002). Stolen-verifier attack on two new strong-password authentication protocols. *IEICE Transactions on Communications*, *vol.E85-B*(11), -25192521.

Chen, T., Lee, W., & Horng, G. (2004). Secure SAS-like password authentication schemes. *Computer Standards & Interfaces*, *27*, 25–31. doi:10.1016/j.csi.2004.02.004

Colin, B., & Anish, M. (2003). *Protocols for Authentication and Key Establishment*. Germany: Springer-Verlag Berlin Heidelberg.

Evans, D., Bond, P., & Bement, A. (2002). *The Keyed-Hash Message Authentication Code (HMAC)*. Gaithersburg.

Haller, N. M. (1995). *The S/KEY one-time password system.*

ISO. (1999). Information Technology - Security Techniques - Entity Authentication - Part 2:Mechanisms Using Symmetric Encipherment Algorithms ISO/IEC 9798-2. International Standard.

Jeong, J., Chung, M. Y., & Choo, H. (2006). Secure User Authentication Mechanism in Digital Home Network Environments. *Lecture Notes in Computer Science*, *4096*, 345–354.

Jeong, Jongpil, Chung, M. Y., & Choo, H. (2008). Integrated OTP-Based User Authentication Scheme Using Smart Cards in Home Networks. *41st Annual Hawaii International Conference on System Sciences (HICSS 2008)* (pp. 294–294). IEEE. doi:10.1109/HICSS.2008.208

Jo, H.S., & Youn, H. Y. (2005). A Secure User Authentication Protocol Based on One-Time-Password for Home Network. *Computational Science and Its Applications – ICCSA*, 519–528.

Jo, Hea Suk, & Youn, H. Y. (2005). Based on One-Time-Password for Home Network *. *ICCSA* (pp. 519–528). Springer-Verlag.

Kahn, D. (1967). *The Codebreakers*. New York: Macmillan Co.

Kamioka, T., & Shimizu, A. (2001). The examination of the security of SAS one-time password authentication. *IEICE Technical Report, OFS2001-48*, (435), 53–58.

Kerckhoffs, A. (1883). La cryptographie militaire. *Journal des SciencesMilitaires*, *IX*, 161–191.

Kocher, P., Jaffe, J., & Jun, B. B. (1999). Differential power analysis. Proceedings of Advances in. *Cryptology (CRYPTO'99)*.

Kohl, J. T., & Neuman, B. C. (1991). The Evolution of the Kerberos Authentication Service. *EurOpen Conference*. Tromsø: IEEE Computer Society Press.

Krawczyk, H., Bellare, M., & Canett, R. (2004). HMAC: Keyed hashing for message authentication, at. *RFC 2104*. Retrieved from http://www.faqs.org/rfcs/rfc2104.html

Lamport, L. (1981). Password Authentication with Insecure Communication. *Communications of the ACM*, *24*(11), 770–772.

Lee, N., & Chen, J. (2005). Improvement of One-Time Password Authentication Scheme Using. *IEICE TRANS. COMUMUN*, *88-B*(9), 3765–3767. doi:10.1093/ietcom/e88

Lee, N. Y., & Chen, J. C. (2005). Improvement of one-time password authentication scheme using smart card. *IEICE Transaction on Communications*, *88*(9), 3765–3769.

Lin, C., Sun, H., & Hwang, T. (2001). Attacks and solutions on strong-password authentication. *IEICE Transactions and Communications, E84-b*(9), 2622–2627.

Lin, M.-H., & Hang, C.-C. (2004). A secure one-time password authentication scheme with low- computation formobile communications. *ACM SIGOPS Operating Systems Review*, *38*(2), 76–84.

Menezes, A. J., Oorschot, P. C. V., & Vanstone, S. A. (1997). *Handbook of Applied Cryptography*. CRC Press.

Messerges, T. S., Dabbish, E. A., & Sloan, R. H. (2002). Examining smart-card security under the threat of power analysis attacks. *IEEE Transactions on Computers*.

M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., & Ranen, O. (2005). *HOTP: An HMAC-based One-Time Password Algorithm, IETF RFC 4226*. Retrieved from http://www.ietf.org/rfc/rfc4226.txt

Raisul, M., Ibne, M., & Mohd, M. (2012). A Review of Smart Homes — Past , Present , and Future. *IEEE Transactions on Systems, Man, and Cybernetics*, 1–14.

Rose, B. (2001). Home networks: a standard perspective. *IEEE CommunicationMagazine*, *39*(12), 78–85.

Sandirigama, M., Shimizu, A., & Noda, M. T. (2000). Simple and secure password authentication protocol (SAS). *IEICE Transaction on Communications, E83-B*(6), 1363–1363.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *The Bell System Technical Journal*, *27*, 379–423.

Stebila, D., Udupi, P., & Sheueling, C. (2010). Multi-Factor Password-Authenticated Key Exchange. *CRPIT-Information Security*, *105*.

Tsuji, T., Kamioka, T., & Shimizu, A. (2002). Simple and secure password authentication protocol,ver.2 (SAS-2). *IEICE Technical Report, OIS2002-30*, *102*(314), 7–11.

Tsuji, T., & Shimizu, A. (2002a). One-time password authentication protocol against theft attacks. *IEICE Transactions on Communications, 87*, 523–529.

Tsuji, T., & Shimizu, A. (2002b). An impersonation attack on one-time password authentication protocol OSPA. *IEICE Technical Report, ISEC2002-81, 102*(436), 67–72.

Vaidya, B., Park, J. H., Yeo, S.-S., & Rodrigues, J. J. P. C. (2011). Robust one-time password authentication scheme using smart card for home network environment. *Computer Communications*, *34*(3), 326–336. doi:10.1016/j.comcom.2010.03.013

Wu, H., Liu, C., & Chiou, S. (2005). Cryptanalysis of a Secure One-Time Password Authentication Scheme with Low-Communication for Mobile Communications. *International Journal of Network Security*, *1*(2), 74–76.

Yeh, T. C., Shen, H. Y., & Hwang, J. J. (2002). A secure one-time password authentication scheme using smart cards. *IEICE Transaction on Communications, 85*, 2515–2518.

ANEXOS

ANEXO 1: Operaciones acumuladas. Protocolos de Vaidya y Vaidya modificado.

Ejemplo para N=100 (cantidad de sesiones máximas para protocolo de Vaidya et al.)

	Vaidya Modif.		Vaidya	
Vez (n)	Operac. Hash acum. de U en cada SR (Protocolo. Modif.)	U (H) de cada AS	Operac. Hash acum. de U en cada SR (Vaidya et al.)	Difer. Acum. (Vaidya - Prot. Modif.)
1	5	5	200	195
2	9	4	299	290
3	13	4	397	384
4	17	4	494	477
5	21	4	590	569
6	25	4	685	660
7	29	4	779	750
8	33	4	872	839
9	37	4	964	927
10	41	4	1055	1014
11	45	4	1145	1100
12	49	4	1234	1185
13	53	4	1322	1269
14	57	4	1409	1352
15	61	4	1495	1434
16	65	4	1580	1515
17	69	4	1664	1595
18	73	4	1747	1674
19	77	4	1829	1752
20	81	4	1910	1829
21	85	4	1990	1905
22	89	4	2069	1980
23	93	4	2147	2054
24	97	4	2224	2127
25	101	4	2300	2199
26	105	4	2375	2270
27	109	4	2449	2340
28	113	4	2522	2409

29	117	4	2594	2477
30	121	4	2665	2544
31	125	4	2735	2610
32	129	4	2804	2675
33	133	4	2872	2739
34	137	4	2939	2802
35	141	4	3005	2864
36	145	4	3070	2925
37	149	4	3134	2985
38	153	4	3197	3044
39	157	4	3259	3102
40	161	4	3320	3159
41	165	4	3380	3215
42	169	4	3439	3270
43	173	4	3497	3324
44	177	4	3554	3377
45	181	4	3610	3429
46	185	4	3665	3480
47	189	4	3719	3530
48	193	4	3772	3579
49	197	4	3824	3627
50	201	4	3875	3674
51	205	4	3925	3720
52	209	4	3974	3765
53	213	4	4022	3809
54	217	4	4069	3852
55	221	4	4115	3894
56	225	4	4160	3935
57	229	4	4204	3975
58	233	4	4247	4014
59	237	4	4289	4052
60	241	4	4330	4089
61	245	4	4370	4125
62	249	4	4409	4160
63	253	4	4447	4194
64	257	4	4484	4227
65	261	4	4520	4259
66	265	4	4555	4290
67	269	4	4589	4320
68	273	4	4622	4349
69	277	4	4654	4377
70	281	4	4685	4404
71	285	4	4715	4430
72	289	4	4744	4455
73	293	4	4772	4479

74	297	4	4799	4502
75	301	4	4825	4524
76	305	4	4850	4545
77	309	4	4874	4565
78	313	4	4897	4584
79	317	4	4919	4602
80	321	4	4940	4619
81	325	4	4960	4635
82	329	4	4979	4650
83	333	4	4997	4664
84	337	4	5014	4677
85	341	4	5030	4689
86	345	4	5045	4700
87	349	4	5059	4710
88	353	4	5072	4719
89	357	4	5084	4727
90	361	4	5095	4734
91	365	4	5105	4740
92	369	4	5114	4745
93	373	4	5122	4749
94	377	4	5129	4752
95	381	4	5135	4754
96	385	4	5140	4755
97	389	4	5144	4755
98	393	4	5147	4754
99	397	4	5149	4752
100	401	4	5150	4749

ANEXO 2: Comparación según cantidad de operaciones en cada solicitud de servicios. Protocolos de Vaidya et al. y Vaidya modificado.

Ejemplo para N=100 (cantidad de sesiones máximas para protocolo de Vaidya et al.)

Sandor			Vaidya		
Vez (n)	U (H)	Operac. Hash de U en cada SR (Protocolo. Modif.)	U (H)	Operac. Hash de U en cada SR (Vaidya et al.)	Diferencia V-S
1	5	5	200	200	195
2	9	4	299	99	95
3	13	4	397	98	94
4	17	4	494	97	93
5	21	4	590	96	92
6	25	4	685	95	91
7	29	4	779	94	90
8	33	4	872	93	89
9	37	4	964	92	88
10	41	4	1055	91	87
11	45	4	1145	90	86
12	49	4	1234	89	85
13	53	4	1322	88	84
14	57	4	1409	87	83
15	61	4	1495	86	82
16	65	4	1580	85	81
17	69	4	1664	84	80
18	73	4	1747	83	79
19	77	4	1829	82	78
20	81	4	1910	81	77
21	85	4	1990	80	76
22	89	4	2069	79	75
23	93	4	2147	78	74
24	97	4	2224	77	73
25	101	4	2300	76	72
26	105	4	2375	75	71
27	109	4	2449	74	70
28	113	4	2522	73	69
29	117	4	2594	72	68
30	121	4	2665	71	67
31	125	4	2735	70	66
32	129	4	2804	69	65
33	133	4	2872	68	64
34	137	4	2939	67	63
35	141	4	3005	66	62
36	145	4	3070	65	61

37	149	4	3134	64	60
38	153	4	3197	63	59
39	157	4	3259	62	58
40	161	4	3320	61	57
41	165	4	3380	60	56
42	169	4	3439	59	55
43	173	4	3497	58	54
44	177	4	3554	57	53
45	181	4	3610	56	52
46	185	4	3665	55	51
47	189	4	3719	54	50
48	193	4	3772	53	49
49	197	4	3824	52	48
50	201	4	3875	51	47
51	205	4	3925	50	46
52	209	4	3974	49	45
53	213	4	4022	48	44
54	217	4	4069	47	43
55	221	4	4115	46	42
56	225	4	4160	45	41
57	229	4	4204	44	40
58	233	4	4247	43	39
59	237	4	4289	42	38
60	241	4	4330	41	37
61	245	4	4370	40	36
62	249	4	4409	39	35
63	253	4	4447	38	34
64	257	4	4484	37	33
65	261	4	4520	36	32
66	265	4	4555	35	31
67	269	4	4589	34	30
68	273	4	4622	33	29
69	277	4	4654	32	28
70	281	4	4685	31	27
71	285	4	4715	30	26
72	289	4	4744	29	25
73	293	4	4772	28	24
74	297	4	4799	27	23
75	301	4	4825	26	22
76	305	4	4850	25	21
77	309	4	4874	24	20
78	313	4	4897	23	19
79	317	4	4919	22	18
80	321	4	4940	21	17
81	325	4	4960	20	16

82	329	4	4979	19	15
83	333	4	4997	18	14
84	337	4	5014	17	13
85	341	4	5030	16	12
86	345	4	5045	15	11
87	349	4	5059	14	10
88	353	4	5072	13	9
89	357	4	5084	12	8
90	361	4	5095	11	7
91	365	4	5105	10	6
92	369	4	5114	9	5
93	373	4	5122	8	4
94	377	4	5129	7	3
95	381	4	5135	6	2
96	385	4	5140	5	1
97	389	4	5144	4	0
98	393	4	5147	3	-1
99	397	4	5149	2	-2
100	401	4	5150	1	-3

ANEXO 3: Tabla comparativa de los protocolos de Vaidya y Vaidya modificado por el autor, en función de las operaciones Hash y XOR en HG y U.

Ejemplo para N=30

	Prot. Modif.				Vaidya				Total HG		Total U		Total (H)		Total (XOR)		Total Operac.			
Vez (n)	HG (H)	HG (XOR)	U (H)	U (XOR)	HG (H)	HG (XOR)	U (H)	U (XOR)	Prot. Modif.	Vaidya	Prot. Modif.	Vaidya	Prot. Modif.	Vaidya	Prot. Modif.	Vaidya	Prot. Modif.	Vaidya	Diferencia Tot. Op.Acum. (Vaidya - Prot. Modif.)	Diferencial V-VM
1	7	6	5	4	104	5	200	4	13	108	9	204	12	304	10	9	22	313	291	
2	12	11	9	8	106	9	299	7	23	114	17	306	21	405	19	16	40	421	381	90
3	17	16	13	12	108	13	397	10	33	120	25	407	30	505	28	23	58	528	470	89
4	22	21	17	16	110	17	494	13	43	126	33	507	39	604	37	30	76	634	558	88
5	27	26	21	20	112	21	590	16	53	132	41	606	48	702	46	37	94	739	645	87
6	32	31	25	24	114	25	685	19	63	138	49	704	57	799	55	44	112	843	731	86
7	37	36	29	28	116	29	779	22	73	144	57	801	66	895	64	51	130	946	816	85
8	42	41	33	32	118	33	872	25	83	150	65	897	75	990	73	58	148	1048	900	84
9	47	46	37	36	120	37	964	28	93	156	73	992	84	1084	82	65	166	1149	983	83
10	52	51	41	40	122	41	1055	31	103	162	81	1086	93	1177	91	72	184	1249	1065	82
11	57	56	45	44	124	45	1145	34	113	168	89	1179	102	1269	100	79	202	1348	1146	81
12	62	61	49	48	126	49	1234	37	123	174	97	1271	111	1360	109	86	220	1446	1226	80
13	67	66	53	52	128	53	1322	40	133	180	105	1362	120	1450	118	93	238	1543	1305	79
14	72	71	57	56	130	57	1409	43	143	186	113	1452	129	1539	127	100	256	1639	1383	78
15	77	76	61	60	132	61	1495	46	153	192	121	1541	138	1627	136	107	274	1734	1460	77
16	82	81	65	64	134	65	1580	49	163	198	129	1629	147	1714	145	114	292	1828	1536	76
17	87	86	69	68	136	69	1664	52	173	204	137	1716	156	1800	154	121	310	1921	1611	75
18	92	91	73	72	138	73	1747	55	183	210	145	1802	165	1885	163	128	328	2013	1685	74
19	97	96	77	76	140	77	1829	58	193	216	153	1887	174	1969	172	135	346	2104	1758	73
20	102	101	81	80	142	81	1910	61	203	222	161	1971	183	2052	181	142	364	2194	1830	72
21	107	106	85	84	144	85	1990	64	213	228	169	2054	192	2134	190	149	382	2283	1901	71

22	112	111	89	88	146	89	2069	67	223	234	177	2136	201	2215	199	156	400	2371	1971	70
23	117	116	93	92	148	93	2147	70	233	240	185	2217	210	2295	208	163	418	2458	2040	69
24	122	121	97	96	150	97	2224	73	243	246	193	2297	219	2374	217	170	436	2544	2108	68
25	127	126	101	100	152	101	2300	76	253	252	201	2376	228	2452	226	177	454	2629	2175	67
26	132	131	105	104	154	105	2375	79	263	258	209	2454	237	2529	235	184	472	2713	2241	66
27	137	136	109	108	156	109	2449	82	273	264	217	2531	246	2605	244	191	490	2796	2306	65
28	142	141	113	112	158	113	2522	85	283	270	225	2607	255	2680	253	198	508	2878	2370	64
29	147	146	117	116	160	117	2594	88	293	276	233	2682	264	2754	262	205	526	2959	2433	63
30	152	151	121	120	162	121	2665	91	303	282	241	2756	273	2827	271	212	544	3039	2495	62
31	157	156	125	124	164	125	2735	94	313	288	249	2829	282	2899	280	219	562	3118	2556	61
32	162	161	129	128	166	129	2804	97	323	294	257	2901	291	2970	289	226	580	3196	2616	60
33	167	166	133	132	168	133	2872	100	333	300	265	2972	300	3040	298	233	598	3273	2675	59
34	172	171	137	136	170	137	2939	103	343	306	273	3042	309	3109	307	240	616	3349	2733	58
35	177	176	141	140	172	141	3005	106	353	312	281	3111	318	3177	316	247	634	3424	2790	57
36	182	181	145	144	174	145	3070	109	363	318	289	3179	327	3244	325	254	652	3498	2846	56
37	187	186	149	148	176	149	3134	112	373	324	297	3246	336	3310	334	261	670	3571	2901	55
38	192	191	153	152	178	153	3197	115	383	330	305	3312	345	3375	343	268	688	3643	2955	54
39	197	196	157	156	180	157	3259	118	393	336	313	3377	354	3439	352	275	706	3714	3008	53
40	202	201	161	160	182	161	3320	121	403	342	321	3441	363	3502	361	282	724	3784	3060	52
41	207	206	165	164	184	165	3380	124	413	348	329	3504	372	3564	370	289	742	3853	3111	51
42	212	211	169	168	186	169	3439	127	423	354	337	3566	381	3625	379	296	760	3921	3161	50
43	217	216	173	172	188	173	3497	130	433	360	345	3627	390	3685	388	303	778	3988	3210	49
44	222	221	177	176	190	177	3554	133	443	366	353	3687	399	3744	397	310	796	4054	3258	48
45	227	226	181	180	192	181	3610	136	453	372	361	3746	408	3802	406	317	814	4119	3305	47
46	232	231	185	184	194	185	3665	139	463	378	369	3804	417	3859	415	324	832	4183	3351	46
47	237	236	189	188	196	189	3719	142	473	384	377	3861	426	3915	424	331	850	4246	3396	45
48	242	241	193	192	198	193	3772	145	483	390	385	3917	435	3970	433	338	868	4308	3440	44
49	247	246	197	196	200	197	3824	148	493	396	393	3972	444	4024	442	345	886	4369	3483	43
50	252	251	201	200	202	201	3875	151	503	402	401	4026	453	4077	451	352	904	4429	3525	42
51	257	256	205	204	204	205	3925	154	513	408	409	4079	462	4129	460	359	922	4488	3566	41
52	262	261	209	208	206	209	3974	157	523	414	417	4131	471	4180	469	366	940	4546	3606	40

53	267	266	213	212	208	213	4022	160	533	420	425	4182	480	4230	478	373	958	4603	3645	39
54	272	271	217	216	210	217	4069	163	543	426	433	4232	489	4279	487	380	976	4659	3683	38
55	277	276	221	220	212	221	4115	166	553	432	441	4281	498	4327	496	387	994	4714	3720	37
56	282	281	225	224	214	225	4160	169	563	438	449	4329	507	4374	505	394	1012	4768	3756	36
57	287	286	229	228	216	229	4204	172	573	444	457	4376	516	4420	514	401	1030	4821	3791	35
58	292	291	233	232	218	233	4247	175	583	450	465	4422	525	4465	523	408	1048	4873	3825	34
59	297	296	237	236	220	237	4289	178	593	456	473	4467	534	4509	532	415	1066	4924	3858	33
60	302	301	241	240	222	241	4330	181	603	462	481	4511	543	4552	541	422	1084	4974	3890	32
61	307	306	245	244	224	245	4370	184	613	468	489	4554	552	4594	550	429	1102	5023	3921	31
62	312	311	249	248	226	249	4409	187	623	474	497	4596	561	4635	559	436	1120	5071	3951	30
63	317	316	253	252	228	253	4447	190	633	480	505	4637	570	4675	568	443	1138	5118	3980	29
64	322	321	257	256	230	257	4484	193	643	486	513	4677	579	4714	577	450	1156	5164	4008	28
65	327	326	261	260	232	261	4520	196	653	492	521	4716	588	4752	586	457	1174	5209	4035	27
66	332	331	265	264	234	265	4555	199	663	498	529	4754	597	4789	595	464	1192	5253	4061	26
67	337	336	269	268	236	269	4589	202	673	504	537	4791	606	4825	604	471	1210	5296	4086	25
68	342	341	273	272	238	273	4622	205	683	510	545	4827	615	4860	613	478	1228	5338	4110	24
69	347	346	277	276	240	277	4654	208	693	516	553	4862	624	4894	622	485	1246	5379	4133	23
70	352	351	281	280	242	281	4685	211	703	522	561	4896	633	4927	631	492	1264	5419	4155	22
71	357	356	285	284	244	285	4715	214	713	528	569	4929	642	4959	640	499	1282	5458	4176	21
72	362	361	289	288	246	289	4744	217	723	534	577	4961	651	4990	649	506	1300	5496	4196	20
73	367	366	293	292	248	293	4772	220	733	540	585	4992	660	5020	658	513	1318	5533	4215	19
74	372	371	297	296	250	297	4799	223	743	546	593	5022	669	5049	667	520	1336	5569	4233	18
75	377	376	301	300	252	301	4825	226	753	552	601	5051	678	5077	676	527	1354	5604	4250	17
76	382	381	305	304	254	305	4850	229	763	558	609	5079	687	5104	685	534	1372	5638	4266	16
77	387	386	309	308	256	309	4874	232	773	564	617	5106	696	5130	694	541	1390	5671	4281	15
78	392	391	313	312	258	313	4897	235	783	570	625	5132	705	5155	703	548	1408	5703	4295	14
79	397	396	317	316	260	317	4919	238	793	576	633	5157	714	5179	712	555	1426	5734	4308	13
80	402	401	321	320	262	321	4940	241	803	582	641	5181	723	5202	721	562	1444	5764	4320	12
81	407	406	325	324	264	325	4960	244	813	588	649	5204	732	5224	730	569	1462	5793	4331	11
82	412	411	329	328	266	329	4979	247	823	594	657	5226	741	5245	739	576	1480	5821	4341	10
83	417	416	333	332	268	333	4997	250	833	600	665	5247	750	5265	748	583	1498	5848	4350	9

84	422	421	337	336	270	337	5014	253	843	606	673	5267	759	5284	757	590	1516	5874	4358	8
85	427	426	341	340	272	341	5030	256	853	612	681	5286	768	5302	766	597	1534	5899	4365	7
86	432	431	345	344	274	345	5045	259	863	618	689	5304	777	5319	775	604	1552	5923	4371	6
87	437	436	349	348	276	349	5059	262	873	624	697	5321	786	5335	784	611	1570	5946	4376	5
88	442	441	353	352	278	353	5072	265	883	630	705	5337	795	5350	793	618	1588	5968	4380	4
89	447	446	357	356	280	357	5084	268	893	636	713	5352	804	5364	802	625	1606	5989	4383	3
90	452	451	361	360	282	361	5095	271	903	642	721	5366	813	5377	811	632	1624	6009	4385	2
91	457	456	365	364	284	365	5105	274	913	648	729	5379	822	5389	820	639	1642	6028	4386	1
92	462	461	369	368	286	369	5114	277	923	654	737	5391	831	5400	829	646	1660	6046	4386	0
93	467	466	373	372	288	373	5122	280	933	660	745	5402	840	5410	838	653	1678	6063	4385	-1
94	472	471	377	376	290	377	5129	283	943	666	753	5412	849	5419	847	660	1696	6079	4383	-2
95	477	476	381	380	292	381	5135	286	953	672	761	5421	858	5427	856	667	1714	6094	4380	-3
96	482	481	385	384	294	385	5140	289	963	678	769	5429	867	5434	865	674	1732	6108	4376	-4
97	487	486	389	388	296	389	5144	292	973	684	777	5436	876	5440	874	681	1750	6121	4371	-5
98	492	491	393	392	298	393	5147	295	983	690	785	5442	885	5445	883	688	1768	6133	4365	-6
99	497	496	397	396	300	397	5149	298	993	696	793	5447	894	5449	892	695	1786	6144	4358	-7
100	502	501	401	400	302	401	5150	301	1003	702	801	5451	903	5452	901	702	1804	6154	4350	-8