Universidad Autónoma del Estado Hidalgo

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

Kerberos

MONOGRAFÍA

QUE PARA OBTENER EL TÍTULO DE
LICENCIADO EN SISTEMAS COMPUTACIONALES

P R E S E N T A

Erick Olvera Hernández

Asesor: **Lic Norma Laura Salazar Viveros**
Coasesor: **Ing. Ana Luisa Vargas Ramírez**

AGRADECIMIENTOS

Este trabajo está dedicado principalmente a la memoria de mi padre y a lo mejor que me paso en la vida que ha sido mi madre, por todo el esfuerzo que realizaron a lo largo de mis estudio, por haber estado siempre presentes apoyándome para que sigua adelante. Agradezco a mis hermanos que siempre me alentaron a seguir

Al igual quiero agradecer a mis amigos por haber estado conmigo en todo momento, por haber creído en mi; Charly, Nacho, Alfonso, Dante, José Luis, Iván, Mauricio, Héctor y todos aquellos que siempre me apoyaron.

Y finalmente mi agradecimiento a todos mis profesores que ayudaron a mi formación profesional, principalmente a mis asesoras la Lic. Norma Laura Salazar Viveros y la Ing. Ana Luisa Vargas Ramírez.

¡Gracias a todos!

ÍNDICE

JUSTIFICACIÓN

Esta investigación es importante en el ámbito de la seguridad en redes, ya que como es sabido Internet es un lugar inseguro, cuando se trata de obtener información de cualquier tipo, la primera opción es Internet. Los usuarios pretenden obtener información precisa, confiable y sobre todo segura, pero muchos de los protocolos que se utilizan actualmente carecen de seguridad necesaria. Además existen *Crackers* y *Hackers* que con frecuencia interceptan contraseñas (*passwords*) o la información que se está transfiriendo entre estaciones de trabajo. De esa manera las aplicaciones que mandan una contraseña no encriptada en la red son extremadamente vulnerables. En el peor de los casos, algunas aplicaciones cliente-servidor asumen que el cliente proveerá su identificación correcta, y otras confían en que el cliente restringirá sus actividades a aquellas que está autorizado.

Algunos sitios utilizan *firewalls* para solucionar los problemas de seguridad en redes, pero los *firewalls* asumen que los ataques provienen “del lado de afuera”, lo cual no es siempre cierto.

Tomando en cuenta que la mayoría de los sistemas de cómputo actuales proporcionan servicios a usuarios múltiples, se requiere la habilidad de validar y autentificación al usuario que está realizando la solicitud. Por ello surge como respuesta los sistemas de autentificación como una solución al problema de la seguridad en las redes, ya que utiliza fuertemente la criptografía, por tanto un cliente puede demostrar su identidad a un servidor y viceversa a través de una conexión insegura.

Dentro de los sistemas de autentificación y distribución de claves más importantes encontramos los siguientes; **Kerberos**, **NetSP**, **SPX**, **TESS**, **SESAME** y **ASF DCE** investigar [OPPLIGER, 1988], [CHESWICK; BELLOVIN, 1994].

Los sistemas de autentificación y de distribución de claves no proporcionan el mismo conjunto de servicios de seguridad por ello se definirá los servicios que proporcionan cada uno de los sistemas de autentificación.

NetSp proporciona sistema de autentificación, confidencialidad en los datos, servicios de integridad. Utiliza funciones *hash* unidireccional con criptografía con clave secreta y ofrece versiones comerciales del sistema

SPX provee de servicio de autentificación, la confidencialidad y servicios de integridad. Maneja funciones *hash* unidireccionales utilizando criptografía con clave secreta, y sus versiones son del dominio público.

TESS ofrece servicio de autentificación, confidencialidad de datos, servicios de integridad y servicios de no rechazo. Opera con funciones *hash* unidireccionales utilizando criptografía con clave secreta pero a diferencia de los anteriores este utiliza también la criptografía con clave pública, y en este sistema todavía no se ha declarado la disponibilidad del sistema.

SESAME suministra los servicios de autentificación, confidencialidad de datos y servicios de integridad al igual que servicios de control de acceso. Utiliza funciones *hash* unidireccionales utilizando criptografía con clave secreta y con clave pública, existen versiones comerciales las versiones son del dominio público.

OSF DCE ofrece los servicios de autentificación, confidencialidad de datos y servicios de integridad al igual que servicios de control de acceso. Emplea funciones *hash* unidireccionales y al igual de los dos anteriores sistemas utilizan la criptografía con clave secreta y con clave pública. Con respecto a su disponibilidad encontramos que no es del dominio público sin embargo existen versiones comerciales.

Por último encontramos el sistema **Kerberos** el cual proporciona los servicios de autentificación, confidencialidad de datos y servicios de integridad. Emplea funciones hash unidireccionales utilizando la criptografía con clave privada. Con respecto a la disponibilidad de los sistemas de autentificación, es importante que algunos sistemas sean de dominio público, mientras que otros se comercializan en el mercado, y podemos decir que el MIT ha hecho del dominio público en Internet las versiones V4 y V5 de **Kerberos**. Pero, además, algunas empresas estuvieron involucradas en el proyecto **Athena** desde sus primeros comienzos y también han desarrollado sus propias versiones de Kerberos, las cuales se vende actualmente como productos comerciales [OPPLIGER, 1988].

Kerberos tiene las siguientes ventajas con respecto a los otros sistemas de autentificación:

- Seguro: Un "*sniffer*" en la red no puede obtener suficiente información necesaria como para hacerse pasar por otro usuario.
- Conveniente: Ya que los servicios a los que autentifica Kerberos no pueden identificar usuarios sin él, Kerberos debe ser lo suficientemente estable como para resistir ataques y deficiencias. Sin duda Kerberos cumple eso.
- Transparente: El usuario no debe enterarse que existe un proceso de identificación más que ingresando alguna que otra *password*.
- Escalable: Debido a las características del sistema, es imprescindible poder crecer en la cantidad de servidores Kerberos y servicios que utilizan Kerberos sin perder seguridad.

Debido a lo anterior y a las ventajas sobre los otros sistemas, y debido a que fue el pionero de los sistemas de autenticación para sistemas en red, y muchos otros diseñados posteriormente basados en mayor o menor medida en **Kerberos**, se convierte en un sistema ideal para investigar [OPPLIGER, 1988], [CHESWICK; BELLOVIN, 1994].

OBJETIVOS

OBJETIVO GENERAL

Proporcionar un conocimiento más profundo de **Kerberos** como solución para seguridad de las redes, las principales ventajas y desventajas ante las redes inseguras, como el Internet, a través de las herramientas de autentificación y criptografía.

OBJETIVOS ESPECÍFICOS

- Definir la encriptación de acuerdo a sus bases etimológicas y de acuerdo los diferentes autores.
- Describir la evolución de la criptografía de manera breve.
- Especificar las diferentes tipos y técnicas de encriptación debido a lo extenso que puede llegar a ser el tema, ya que día a día se siguen desarrollando nuevas tecnologías y métodos de encriptación.
- Definir de manera clara lo que es **Kerberos** así como su desarrollo.
- Describir el protocolo de *Needham – Shchoreder*, ya que este protocolo es la base de **Kerberos**.
- Describir de manera detallada la arquitectura del sistema de autentificación de **Kerberos**.
- Plantear los métodos de criptografía que utiliza **Kerberos**.
- Describir de manera individualizada los diversos pasos en el proceso de autentificación de **Kerberos**.
- Detallar las diferentes versiones de **Kerberos**.
- Definir y describir la autentificación entre reinos.
- Plantear los problemas que presenta en las diversas versiones **Kerberos**.

INTRODUCCIÓN

La criptografía es un una ciencia con más de 2000 años de antigüedad, surge por la necesidad de mantener comunicaciones confiables sobre canales inseguros, es decir, donde dos entidades puedan intercambiar información mediante un canal que puede ser interceptado por una tercera entidad no autorizada para leer la información, a partir de su invención y uso, se ha podido enviar los mensajes con mayor seguridad y confiabilidad.

En la actualidad se realizan grandes transacciones comerciales y financieras, se intercambian una gran cantidad de datos, se comparten recursos, etc., debido a la evolución continua que mantienen los sistemas de cómputo. Conforme a los avances de las telecomunicaciones se requiere un mecanismo que otorgue seguridad a la información, protegiéndola de personas, entidades o procesos no autorizados. Este mecanismo es la criptografía, el cual no es más que una máscara de la información para mantenerla segura.

Las redes modernas van por todas partes, incluso las redes pequeñas a menudo conectan a un laboratorio a un sitio de trabajo, las más grandes conectan a continentes múltiples incluso de una misma compañía, si se lograra conectar un dispositivo de tal red, se podría tener acceso a cualquier punto de dicha red y sin ningún control de acceso apropiado.

Kerberos fue creado en el MIT *(Massachusetts Institute of Technology)* como una solución al problema en redes. El sistema de autentificación de Kerberos, facilita los medios necesarios para proporcionar la seguridad a los sistemas en red, uno de los medios más importantes es su servicio de autentificación, dicho servicio utiliza la “criptografía”, por tanto un cliente puede demostrar su identidad a un servidor y viceversa en una red considerada insegura. Posteriormente que sean autentificado, se encriptan toda la comunicación para asegurar la privacidad e integridad de los datos. Aún que Kerberos se utiliza comúnmente en sistemas compuestos de estaciones de trabajo y servidores, puede introducirse en sistemas más pequeños

CAPITULO I

CRIPTOGRAFÍA

1.1 INTRODUCCIÓN

El uso de la criptografía así como las técnicas criptográficas tienen como propósito prevenir algunas faltas de seguridad en un sistema computarizado. Actualmente existe una gran cantidad de software y hardware destinado a analizar y monitorizar el tráfico de datos en redes de computadoras; si bien estas herramientas constituyen un avance en técnicas de seguridad y protección, su uso indebido es al mismo tiempo un grave problema y una enorme fuente de ataques a la intimidad de los usuarios y a la integridad de los propios sistemas. En la actualidad no se persigue únicamente la privacidad o confidencialidad de los datos, sino que se busca además garantizar la autenticación de los mismos (el emisor del mensaje es quien dice ser, y no otro), su integridad (el mensaje que se recibe es el mismo que fue enviado) y su no repudio (el emisor no puede negar él haber enviado el mensaje). La seguridad en general debe de ser considerada como un aspecto de gran importancia. El hecho que gran parte de actividades humanas sea cada vez más dependiente de las redes de computadoras hace que la seguridad juegue un papel de suma importancia.

Es importante mencionar algunos datos relacionados con la seguridad antes de comenzar con el desarrollo del tema. De acuerdo con algunos informes dados por el FBI a finales de 1999, las empresas cuentan con diversas tecnologías de seguridad, entre las más comunes se encuentran; controles de acceso, archivos cifrados, sistemas de *password, Firewall,* sistemas de *login* y certificados digitales para la autentificación. Estos informes indican que los principales autores de los ataques a las empresas son *Hackers*, los competidores y los propios empleados. Siempre se podrán encontrar motivos para reafirmar la trascendencia que tiene la seguridad en los sistemas computarizados [ÁNGEL, 1999].

1.2 DEFINICIÓN

La palabra criptografía proviene del griego "**kryptos**" que significa esconder y "**gráphein**" que significa escribir, es decir, **escritura escondida**. "*La criptografía es la ciencia que trata los problemas teóricos relacionados con la seguridad en el intercambio de mensajes en clave entre un emisor y un receptor a través de un canal de comunicaciones (en términos informáticos, ese canal suele ser una red de computadoras)".* La criptografía ha sido usada a través de los años para mandar mensajes confidenciales cuyo propósito es que sólo las personas autorizadas puedan entender el mensaje [PFAFFENBERGER; WALL, 1996].

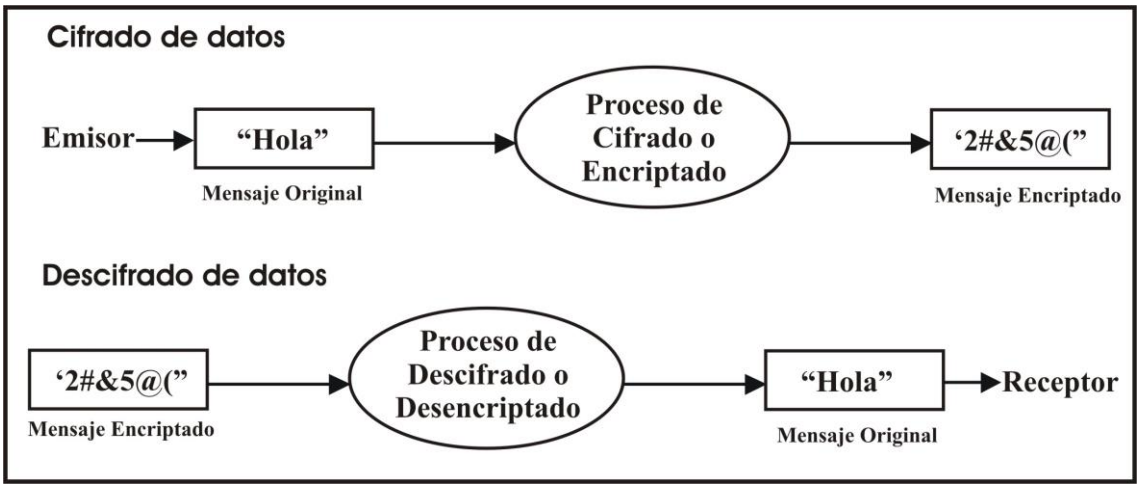

Figura 1.1 Proceso de Encriptación

Como se muestra en la figura 1.1, el emisor del mensaje que envía información confidencial aplicando técnicas criptográficas para "esconder" el mensaje (se define como cifrar o encriptar), esta transmisión se realiza por un canal de comunicaciones (red de computadoras) que se supone insegura y después solo el receptor autorizado pueda leer el mensaje "escondido" (se define como descifrar o descencriptar). Esta ciencia está dividida en dos grandes ramas: la criptografía, ocupada del cifrado de mensajes en clave así como del diseño de criptosistemas y el criptoanálisis, que trata de descifrar los mensajes en clave, rompiendo así el criptosistema [ÁNGEL, 1999], [2].

1.3 EVOLUCIÓN

La criptografía es tan antigua como la escritura misma. Desde que el ***homo - sapiens*** camino sobre este planeta, ha necesitado comunicarse con sus semejantes, pero en ocasiones era importante que otros no se enteren. Las razones son indudables, ya que a nadie le gustaría que el enemigo conociera su estrategia si este último lograse interceptar un mensaje.

En Egipto la criptografía alcanzó la categoría de ciencia, mientras el pueblo utilizaba la lengua demótica, los sacerdotes egipcios hicieron uso con una abundancia de la escritura hierática (jeroglífica) que ha resultado desesperante para los modernos investigadores. En las tumbas del antiguo **Egipto** existen múltiples ejemplos de escritura cifrada, además de su valor práctico, se le atribuía un valor mágico y religioso [1], [2].

Hay quien se atribuye el primer método de encriptación al general romano **Julio César**, quien creó un sistema simple de sustitución de letras, que consistía en escribir el documento codificado con la tercera letra que siguiera a la que realmente correspondía. La A era sustituida por la D, la B por la E y así sucesivamente por consiguiente cada letra del alfabeto es desplazada tres lugares de su posición original. El emperador **Augusto** utilizó un sistema similar, en lugar de sustituir una letra por otra letra situada tres lugares después, la sustituía por la siguiente letra del alfabeto [3], [5].

El primer caso claro de uso de métodos criptográficos se dio durante la guerra entre **Atenas** *y* **Esparta**, los ***éforos*** (gobernantes) espartanos transmitían sus instrucciones a sus *estrategas* (generales) utilizando un bastón, el *escitalo*. El historiador griego **Plutarco** describe el *escitalo* o *scitala* espartana como una vara de la que se preparaban dos o más bastones idénticos. Las órdenes se escribían en una tira de pergamino o papiro enrollada a lo largo del bastón. Desenrollada, solamente contenía una sucesión de letras inconexas que se enviaba al

destinatario, para leer el mensaje éste debía tener en su poder una copia del bastón. Al colocar de nuevo la cinta en el bastón aparecía el mensaje como se muestra en la figura 1.2, se observa el texto AVANZAR DURANTE DOS DIAS HACIA EL RIO o lo que es lo mismo: ARTIIO VEDEAA AUDSE NROHL ZASAR ANDCI [CABALLERO, 1996], [1], [3], [8].

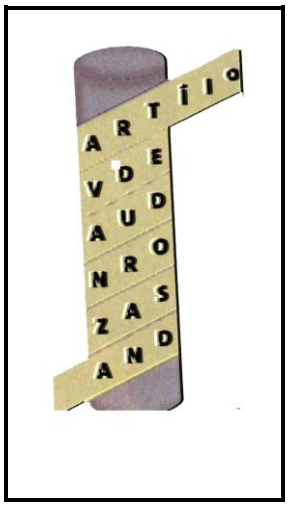

Figura 1.2 Escitalo

Antiguos textos judíos fueron encriptados siguiendo el método de sustituir la primera letra del alfabeto por la última y así sucesivamente. En la **Biblia** (**Jeremías 25:26**) el nombre de **Babilonia** aparece encriptado como "*Sheshech*".

Otro sistema conocido como *atbash* hebreo citado en la **Biblia**, concretamente en el libro de Jeremías, era una forma simple de codificación en donde las letras del alfabeto se colocaban en dos columnas. La superior está escrita en sucesión de izquierda a derecha y la inferior de derecha a izquierda. Si la letra a encriptar aparece en la primera fila se sustituye por la inmediata inferior; y al revés, si aparece abajo, por la inmediata superior [3].

Pero fue el abate español ***Tritemio (Johannes Trithemius),*** quien con su obra **"*Polygraphiae libri sex"***, dio gran impulso a la criptografía. Nacido en

Trittenheim, una pequeña ciudad alemana de la que le viene el nombre, en 1462, era un docto abad benedictino que se interesó por las ciencias naturales y se ganó fama de mago. Ideó un cuadrado en el que en cada línea aparecía el alfabeto, pero la línea inferior empezaba una letra después, y repetía el ciclo. Este cuadrado se le conoce actualmente -erróneamente- como cuadrado de ***Vigenère*** [1].

León Battista Alberti (1402-1472), uno de los más famosos exponentes del Renacimiento italiano, escribió un tratado titulado "***Modus scribendi in jiferas***", donde describe, entre otras cosas, unos discos con los que se podían cifrar mensajes. Como se observa en la figura 1.3, este sistema está compuesto por dos discos uno interior y otro exterior. En el disco externo hay escritos números y letras. En el interior aparecen los signos cifrados, para crear una clave se gira el externo y se hace corresponder la letra "M" a otra preestablecida, en el ejemplo una "t", una "o" ó una "h" permite cambiar la clave del mensaje constantemente, impidiendo que sea descifrado [1], [8].

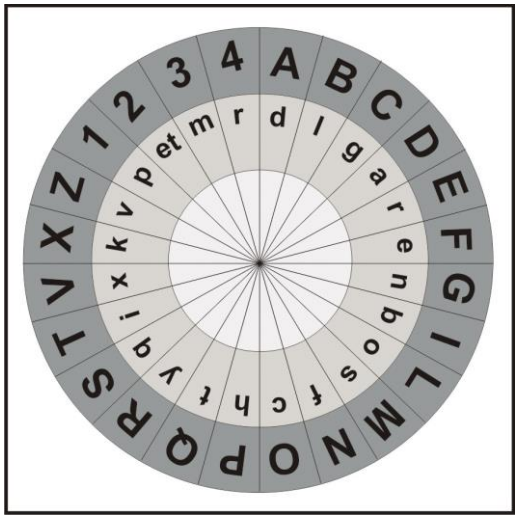

Figura 1.3 Círculos Concéntricos

Sin embargo se considera al abate ***Johannes Trithemius*** como padre de la criptografía moderna. Este sistema está compuesto por dos discos uno interior y otro exterior. [1].

Carlos I de **Inglaterra** usaba en el siglo XVII códigos de sustitución silábica. Napoleón, en sus campañas militares y en los escritos diplomáticos usaba los llamados métodos ***Richelieu*** y ***Rossignol***, para evitar la regularidad de los símbolos asignaba números a grupos de una o más letras. El siglo XVIII no fue una época de grandes avances criptográficos. El telégrafo, inventado por **Samuel Morse** a principios del siglo XIX, y la aparición de la radio cambiaron las comunicaciones, obligando a la criptografía a desarrollarse como ciencia. Ambos medios de comunicación eran fáciles de interceptar lo que dio lugar a que en los ámbitos militares y diplomáticos se buscaran nuevas maneras de mantener en secreto mensajes importantes [CABALLERO, 1996], [1], [2].

Algunos escritores famosos han utilizado la criptografía en sus novelas, tal es el caso de ***Edgar Allan Poe*** un excelente criptoanalista aficionado, publicó en 1843 un famoso relato titulado" El escarabajo de oro" en el que se narra la aventura de un individuo que encuentra un mensaje cifrado donde se indica la localización de un tesoro. También ***Julio Verne*** (1828-1905) utilizó la criptografía en dos de sus novelas, concretamente en "**Viaje al centro de la Tierra**", "***Mathias Sandorf***" [1].

Durante las dos guerras mundiales, la criptografía adquirió un alto grado de perfección, especialmente en la última guerra. El amplio uso, por todos los escalones del mando, de los medios de telecomunicación como eran el teléfono, telégrafo y radio, con el constante peligro de que las conferencias y mensajes pudieran ser interceptados por los enemigos, se hizo indispensable que las fuerzas armadas utilizaran el lenguaje cifrado y mantuvieran la renovación continua de las claves [CABALLERO,1996], [8].

Durante la Primera Guerra Mundial, el trasatlántico ***Lusitania*** fue hundido por los alemanes en 1915, pereciendo más de 1,200 personas. Este hecho polarizó la opinión pública norteamericana apoyando la entrada de los Estados Unidos en la guerra. El presidente ***T. Woodrow Wilson***, quien había fomentado activamente la

no beligerancia de su país, se oponía. Sin embargo, el 17 de enero de 1917 los servicios secretos ingleses interceptaron un telegrama cifrado del ministro de Asuntos Exteriores de **Alemania**, ***Arthur Zimmermann***, dirigido al conde ***Heinrich A. Von Bernstorff***, embajador alemán en **Washington**.

El mensaje se había cifrado por medio del código 0075, un diccionario de lista doble de 10000 palabras o frases que los ingleses ya habían conseguido descifrar parcialmente. En él se hacía referencia a otro telegrama que debía enviarse a **México**. Los ingleses consiguieron interceptar este segundo telegrama y observaron que se había cifrado con otro código, el 13040, que constaba de 25000 palabras, esto representó un grave error criptográfico ya que permitió a los ingleses descifrar el mensaje. El resultado fue sorprendente: en dicho telegrama se le proponía a **México** declararle la guerra a los **Estados Unidos** y en caso de resultar vencedores **Alemania** apoyaría a **México** para recuperar sus antiguos territorios de **Nuevo México**, **Arizona** y **Texas**. *Wilson* se vio forzado a entrar en guerra contra Alemania [1], [2], [3].

En 1938, un año antes del inicio de la Segunda Guerra Mundial, la inteligencia británica reunió en ***Bletchley Park***, a 80 kilómetros al noroeste de Londres, a un reducido grupo de criptoanalistas, naciendo así la "**Estación X**". La misión de este grupo era romper la cifra de la máquina de codificación nazi "**Enigma**" que se muestra en la figura 1.4. Inventada en 1918 para transacciones bancarias, "**Enigma**" era utilizada por el partido nazi para cifrar la mayor parte de sus comunicaciones secretas, de hecho los alemanes habían aumentado la complejidad de su diseño y la clave de cifrado era cambiada diariamente [ÁNGEL, 1999], [1].

Figura 1.4 Maquina Enigma

En ***Bletchley Park*** trabajaba uno de los matemáticos más brillantes del siglo, uno de los más famosos fue ***Alan Turing***. Él creó las bombas; dispositivos electromecánicos que reducían el tiempo necesario para encontrar la clave del día. Estas bombas constituyeron la antesala al desarrollo de la primera computadora del mundo, "***Colossus***" que se muestra en la figura 1.5.

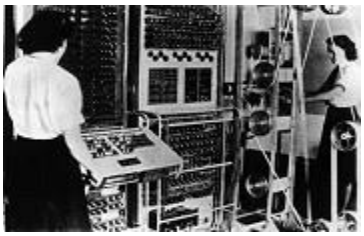

Figura 1.5 Colossus

Por su parte, la marina norteamericana se enfrentó al código japonés *JN-25*. Para la primavera de 1942 el código estaba parcialmente roto y se comenzaron a interceptar muchos mensajes refiriéndose a un objetivo designado como "AF".

Los oficiales de inteligencia del almirante ***Chester Nimitz***, comandante en jefe de la flota del Pacífico, sospechaban que se referían a ***Midway***. Para estar seguros difundieron una información falsa sobre un problema con la planta de destilación de agua de la isla. Y llegó la confirmación. Un mensaje codificado japonés decía: "AF está escaso de agua". Los japoneses iban a atacar ***Midway*** y allí les estarían esperando. Esta batalla cambió el curso de la guerra en el Pacífico: los japoneses

tuvieron que retirarse. Aunque los códigos norteamericanos fueron rotos también en numerosas ocasiones, hubo uno, utilizado por la Marina, que jamás fue descifrado.

Los americanos emplearon un sistema de criptofonía muy original, el lenguaje navajo. Por su sintaxis y cualidades tonales, este lenguaje no escrito resulta ininteligible si no se ha recibido un intenso aprendizaje. Los marines utilizaban indios navajos para transmitir los mensajes por teléfono. Se les daba el mensaje a transmitir en inglés americano y ellos lo transmitían en su lengua, el navajo. Otro indio navajo al otro lado de la línea escuchaba el mensaje y lo traducía de nuevo al inglés americano. Los navajos podían codificar, transmitir y decodificar tres líneas de mensaje en 20 segundos como se puede observar en la figura 1.6 [ÁNGEL, 1999], [1], [3].

Figura 1.6 Radioperador Navajo

Hasta la Segunda Guerra Mundial, todos los cifrados eran de clave secreta o simétrica, a finales de los años 40, el precursor en la teoría de la información, ***C. E. Shannon***, sugirió el uso de una mezcla de transposiciones y sustituciones. Ya en los 70's, IBM retomó el consejo de ***Shannon*** y desarrolló un sistema al que llamo **LUCIFER**. En 1976, el gobierno de los EU adoptó una variante de este sistema, el DES (*Data Encryption Standard*), la principal ventaja es él usar claves aleatorias lo que dificulta el criptoanálisis. Sin embargo, como emisor y receptor deben conocer la clave, ésta, al ser enviada, aún es susceptible de ser interceptada [ÁNGEL, 1999], [1].

En 1975, en la Universidad de *Stanford, dos ingenieros;* ***Whitfield Diffie*** *y* ***Martín Hellman*** comenzaron una gran revolución, al crear la criptografía de clave pública. La nueva idea consistía en algo aparentemente imposible: un criptosistema en donde hubiera dos claves, una para cifrar y otra para descifrar. El sistema está basado en que si dos personas quieren intercambiar mensajes cada uno crea su pareja de claves y hace pública la clave de cifrado. Entonces la persona 1, haciendo uso de la clave de cifrado de la persona 2, compone y envía su mensaje, la persona 2 al recibirlo podrá descencriptar haciendo uso de su clave para descifrar. De la misma manera la persona 2 podrá enviar mensajes cifrados a la persona 1 haciendo uso de la clave de cifrado que este le proporcionó [4].

En la actualidad, el método de clave pública más utilizada es llamado RSA (de ***Rivest***, ***Shamir*** *y* ***Adleman***), desarrollado en 1977 por ***Adi Shamir, Ronald Rivest*** *y* ***Leonard Adleman*** que aparecen en la figura 1.7. Se basa en que no existe un algoritmo lo suficientemente eficiente para factorizar grandes números que sean producto de dos números primos, su desventaja es la velocidad, mientras DES encripta a una velocidad de 20 Mb/seg, RSA lo hace a razón de 20 Kb/seg [ÁNGEL, 1999], [1], [3].

Figura 1.7 ***Shamir / Rivest / Adleman***

1.4 UTILIDADES DE LA CRIPTOGRAFÍA

A través de la criptografía se puede evitar que un documento digital que ha sido transmitido por un canal, sin protección alguna, sea modificado intencionadamente [CABALLERO, 1996], [ÁNGEL, 1999].

- Se garantiza el origen de un documento, es decir, puede verificar que el documento proviene de quien dice provenir.
- Se garantiza la autenticidad del documento, esto significa que no ha sido modificado por ninguna persona.
- Se verifica que el receptor es la entidad correcta a quien va dirigido el documento, es decir, se consigue evitar el entregar un documento a un impostor.
- La criptografía ofrece seguridad en las comunicaciones digitales, siempre que se aplique de una forma correcta [2].

1.5 CRIPTOSISTEMA

Es importante definir a que se refiere un cripotosistema por la mención que se hace sobre ellos en la definición sobre la criptografía.

Matemáticamente, se puede definir un criptosistema como una cuaterna de elementos $\{A, K, \varepsilon, D\}$ formada por [9], [BLACK, 1995]:

- Un conjunto finito llamado **alfabeto**, A, a partir del cual, y utilizando ciertas normas sintácticas y semánticas, se puede emitir un mensaje en claro (*plain text*) u obtener el texto en claro correspondiente a un mensaje cifrado (*cipher text*). Frecuentemente, este alfabeto es el conjunto de los enteros módulo q, Z_q, para un $q \in N$ dado.
- Otro conjunto finito denominado **espacio de claves**, K, formado por todas las posibles claves, tanto de cifrado como de descifrado, del criptosistema.
- Una familia de aplicaciones del alfabeto en sí mismo, $\varepsilon : A \to A$, llamadas **transformaciones de cifrado**. El proceso de cifrado se suele representar como

$$\varepsilon\ (k,a) = c \text{,donde } k \in K \quad a \in A \text{ y } \quad c \in A$$

- Otra familia de aplicaciones del alfabeto en sí mismo, $D: A \to A$, llamadas transformaciones de descifrado. Análogamente al proceso de cifrado, el descifrado se presenta como:

$$D(k', c) = m \text{ Donde } k' \in K \quad c \in A \text{ y } m \in A$$

Casi todos los autores dividen a su vez un miembro de esta cuaterna, el alfabeto, en dos espacios diferentes: el espacio de mensajes, M, formado por los textos en claro que se pueden formar con el alfabeto, y el espacio de cifrados, C, formado por todos los posibles criptogramas que el cifrador es capaz de producir. Sin embargo, tanto el texto en claro como el cifrado han de pertenecer al alfabeto, por lo que no se hacen distinciones entre uno y otro, agrupándolos en el conjunto A para simplificar los conceptos que presentan. [BLACK, 1995].

Así, un criptosistema presenta la estructura mostrada en la figura 1.8, el emisor emite un texto en claro, que es tratado por un cifrador con la ayuda de una cierta clave, k, creando un texto cifrado (criptograma). Este criptograma llega al descifrador a través de un canal de comunicaciones (este canal será habitualmente algún tipo de red). El descifrador convierte el criptograma de nuevo en texto claro, apoyándose ahora en otra clave, k´ y este texto claro ha de coincidir con el emitido inicialmente para que se cumplan los principios básicos de la criptografía moderna. En este hecho radica toda la importancia de los criptosistemas. Es obvio, que el elemento más importante de todo el criptosistema es el cifrador, que ha de utilizar el algoritmo de cifrado para convertir el texto claro en un criptograma. Usualmente, para hacer esto, el cifrador depende de un parámetro exterior, llamado clave de cifrado (o de descifrado) que es aplicado a una función matemática irreversible (al menos, computacionalmente), no es posible invertir la función a no ser que se disponga de la clave de descifrado. De esta forma, cualquier conocedor de la clave (y, por supuesto, de la función), será capaz de descifrar el criptograma, y nadie que no conozca dicha clave puede ser

capaz del descifrado, aún en el caso de que se conozca la función utilizada [ÁNGEL, 1999], [BLACK, 1995], [2], [9].

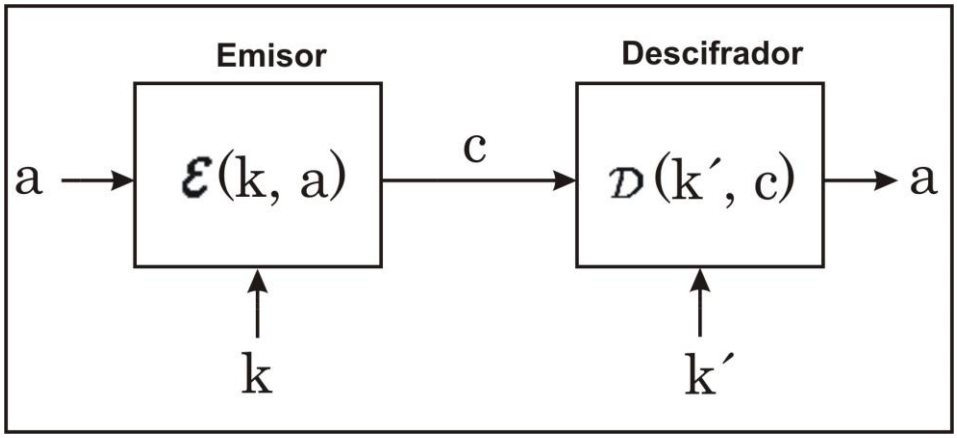

Figura 1.8 Estructura de un Criptosistema

1.6 CLASIFICACIÓN DE LOS ALGORITMOS DE CIFRADO

Con la aparición y posterior desarrollo del cifrado, se han desarrollado diversas formas de protección de la información. El cifrado permite la confidencialidad de los datos y solamente deben ser accesibles a aquellas personas que estén autorizadas a conocer la información, también permite la integridad de los datos, los cuales deben ser legítimos, ya que no serviría de nada datos que hayan podido ser cambiados o no estén completos y por último la disponibilidad, deben estar disponibles para las personas autorizadas.

Diversos autores concuerdan en que los algoritmos de cifrado se pueden dividir en tres grandes grupos:

1.- Según la naturaleza del algoritmo de cifrado [ÁNGEL, 1999], [CABALLERO, 1996] [BLACK, 1995], [2], [3]:

- **Sustitución**: Sustituye unos símbolos por otros.

- **Permutación (Transposición)**: No sustituye los símbolos únicamente cambia el orden de los mismos.

- **Producto (Supercifrado ó Recifrados)**: Son los cifrados obtenidos aplicando 2 o más veces los métodos anteriores, cuantos más métodos se apliquen más seguridad se tiene, siendo este método el más aplicado en la actualidad.

2.- Según la clave [ÁNGEL, 1999], [BLACK, 1995], [CABALLERO, 1996], [2], [3], [4]:

- **Simétricos (clave privada)**: la clave de cifrado, puede ser calculada a partir de la de descifrado, y viceversa. En la mayoría de estos sistemas, ambas claves coinciden, y por supuesto han de mantenerse como un secreto entre emisor y receptor: si un atacante descubre la clave utilizada en la comunicación, ha roto el criptosistema.

- **Asimétricos (Clave Pública)**: la clave de cifrado se hace de conocimiento general (se le llama clave pública). Sin embargo, no ocurre lo mismo con la clave de descifrado (clave privada), que se ha de mantener en secreto. Ambas claves no son independientes, pero del conocimiento de la pública no es posible deducir la privada sin ningún otro dato (en los sistemas de clave privada sucedía lo contrario).

- **Irreversibles**: Cifran un texto no permitiendo su descifrado, una de sus utilizaciones es la del cifrado de contraseñas, otra aplicación es la de las claves desechables o dinámicas que se utilizan en ciertos teléfonos móviles.

3.- Según el número de símbolos cifrados a la vez [ÁNGEL, 1999], [CABALLERO, 1996], [2], [3]:

- **Bloque**: Toman el texto en claro y lo dividen en bloques de igual longitud y cifran cada bloque independientemente. Suelen emplearse bloques de 64 bits.

- **Flujo**: El texto en claro se cifra símbolo tras símbolo, cifrándose cada uno con clave diferente.

1.7 TIPOS DE SECRETOS

En criptografía se definen varios niveles de seguridad en los cuales se pueden enmarcar los diferentes algoritmos criptográficos [BLACK, 1995], [3]:

- **Secreto Perfecto**: El mensaje es seguro frente a tiempo y recursos ilimitados. En este tipo de cifrado el tamaño de la clave es mayor o igual que el tamaño del texto a cifrar.

- **Secreto Computacional**: El mensaje es seguro frente a ataques con tiempo y recursos limitados.

  Ejemplo: Criptosistemas de clave pública
- **Secreto Probable**: El mensaje se encuentra probablemente seguro.
  Ejemplo: Criptosistemas de clave privada

- **Secreto condicional**: La seguridad del mensaje depende de las características de su entorno.
  Ejemplo: Un mensaje no cifrado o cifrado utilizando criptosistemas clásicos, que se envía a través de una red "segura".

CAPITULO II

TÉCNICAS DE ENCRIPTACIÓN

2.1 CIFRADOS CLÁSICOS

Se puede realizar otra gran división de los cifrados según el tipo de operación que se realiza en el cifrado. Dada la característica finita del alfabeto y la hipótesis de no variación de la longitud del texto, existen dos opciones para el cifrado. La primera, llamada **sustitución**; consiste en suplir las unidades del texto original por otras; la segunda llamada **transposición**; consiste en desordenar las unidades que forman el texto original [ÁNGEL, 1999], [CABALLERO, 1996], [BLACK, 1995], [2], [3].

2.1.1 ALGORITMOS DE SUSTITUCIÓN

Este método consiste básicamente en sustituir los caracteres del mensaje inicial por otros, dichos caracteres pueden ser de cualquier tipo: letras, símbolos, dígitos, etc. Los caracteres iníciales continúan estando en el mismo orden, pero solo que se conozca la equivalencia entre los nuevos caracteres y los antiguos, el mensaje es ilegible [2].

Se pueden considerar dos tipos de sustitución [ÁNGEL, 1999], [BLACK, 1995], [CABALLERO, 1996], [HALSALL, 1998], [2], [3]:

- **Sustitución Mono-alfabética** (Equivalencia entre alfabetos carácter a carácter): A cada letra del alfabeto ordinario se le hace corresponder un símbolo y el mensaje se cifra cambiando las letras iníciales por su equivalente, un ejemplo más claro de este tipo de sustitución el cifrado César.

- **Sustitución Poli-alfabética** (Utilización de cifra o clave): Distinto del anterior porque una vez establecida la correspondencia entre alfabetos (que en este caso puede utilizar otro tipo de caracteres), el ejemplo más claro de este tipo de sustitución es el de cifrado *Vigènere*, que se explicara mas delante de este capítulo.

2.1.1.1 CIFRADO DE CESAR

El algoritmo de César o Caesar, llamado así debido a que lo empleaba Julio César para enviar mensajes secretos en que presuntamente lo utilizó para establecer comunicaciones seguras con sus generales durante las Guerras Gálicas, es uno de los algoritmos criptográficos más simples.

El sistema reemplaza cada letra por la situada tres posiciones delante en el alfabeto, dicho de otra manera consiste en sumar 3 al número de orden de cada letra como se muestra en la tabla 2.1 [BLACK, 1995].

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Tabla 2.1 Cifrado César Alfabético

Por ejemplo:

MENSAJE LLEGUE VI Y VENCI

MENSAJE CIFRADO OOHJXH YL B YHQFL

Matemáticamente, para trabajar con el cifrado César o Caesar, se asignan a cada letra un número (A = 0, B = 1. . .), y se considera un alfabeto de 26 letras, la transformación criptográfica sería [2], [6]:

C = (M + 3) mod 26

Asumiendo un alfabeto de 26 símbolos dará como resultado la tabla 2.2:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Tabla 2.2 Cifrado César Matemático

Como ejemplo se cifrara el siguiente mensaje:

ZAPATO

Se puede hacer manualmente ó utilizado la fórmula anteriormente dada:

1. Reemplazar **M** por el valor de la primera letra, en este caso Z equivale a 25.

2. Realizar la operación indicada:

   C = (25 + 3) mod 26 = 2

3. Realizar la operación con las letras restantes.

Mensaje Codificado:

CDSDWR

2.1.1.2 CIFRADO DE *HILL*

Los cifrados monográficos, en los que se suple un carácter por otro de una forma preestablecida, son vulnerables al análisis de frecuencia de aparición de las letras. Para evitarlo se desarrollaron esquemas basados en cifrar bloques de letras de una cierta longitud fija, o sea, cifrado poligráfico. El esquema que aquí se tratara se debe a *Hill* (hacia 1.930 D. C.).

Un cifrado de *Hill* se obtiene al transformar bloques de **n** caracteres en un texto cifrado a través de la relación $\boldsymbol{C = K \cdot P}$ **(mod 28**), donde [BLACK, 1995], [2], [3]:

- K es una matriz n x n, que debe ser inversible módulo 28, es decir, el m.c.d (determinante_de_la_matriz, 28=1).
- P es un bloque de n caracteres.
- C es la matriz columna resultante del cifrado de P.
- 28 es el número de símbolos del alfabeto [ABCDEFGHIJKLMNÑOPQRSTUVWXYZ]

Un ejemplo para un cifrado digráfico (bloques de 2 caracteres) sería para el texto original se muestra en la tabla 2.3.

E	S	T	A	C	I	O	N		C	E	N	T	R	A	L		X
5	20	21	1	3	9	16	14	0	3	5	14	21	19	1	12	0	25

Tabla 2.3 Cifrado Digráfico

Se dispone el texto de la forma en que lo muestra la tabla 2.4 y se aplica la transformación indicada.

E	T	C	O		E	T	A	
S	A	I	N	C	N	R	L	X

Tabla 2.4 Disposición de texto

Se realizan las operaciones:

C1= (1 * P1) + (27 *P2) (mod 28)
C2= (0 * P1) + (3 *P2) (mod 28)

$$K = \begin{pmatrix} 1 & 27 \\ 0 & 3 \end{pmatrix}$$

Donde **P1** y **P2** son dos caracteres del mensaje sin cifrar, **C1** Y **C2** los correspondientes cifrados y **K**.

Continuando con el ejemplo y codificando:

E
S

Tabla 2.5 Caracteres P1y P2

Siendo E = 5 y S = 20, entonces:

C1= (1 * 5) + (27 * 20) = 545 = 13 (mod 28) (letra M)
C2= (0 * 5) + (3 * 20) = 60 = 4 (mod 28) (letra D)

$$\begin{pmatrix} C_1 \\ C_2 \end{pmatrix} = \begin{pmatrix} 1 & 27 \\ 0 & 3 \end{pmatrix} (\text{mod } 28)$$

Y así sucesivamente para cada bloque de 2 caracteres, resultando:

MDSCUZBNXIRNBAPHCR

La consecuencia es que el mismo carácter se codifica de distintas formas (la primera E se ha codificado como una M, y la segunda E del texto original se ha codificado como una D).

El descifrado del sistema de *Hill* es simétrico (la clave de desencriptación se calcula a partir de la clave de encriptación y viceversa) y se debe aplicar la transformación:

$$P = K^{-1} \cdot C \pmod{28}$$, donde K^{-1} es la matriz inversa de K

Para calcular la inversa módulo **n** de una matriz cualquiera:
Si **A** es una matriz tal que m.c.d (det(**A**), **n**)=1 y se llama **d** = det(**A**) entonces **A'** =**d'**. **B** donde **d'** es el inverso de **d** módulo **n** y **B** es la traspuesta de la matriz adjunta de **A** [ÁNGEL, 1999], [BLACK, 1995], [2], [3].

2.1.1.3 CIFRADO *VIGÈNERE*

El sistema de cifrado de *Vigenère* (en honor al criptógrafo francés del mismo nombre) es un sistema poli alfabético o de sustitución múltiple. Este tipo de criptosistemas aparecieron para sustituir a los mono alfabéticos o de sustitución simple, basados en el César, que este ultimo presentaba ciertas debilidades frente a los criptoanálisis debido a la frecuencia con las que aparecían los elementos del alfabeto.

El principal elemento de este sistema es la llamada Tabla de *Vigenère*, una matriz de caracteres cuadrada, con dimensión, que se muestra en la tabla 2.6 [ÁNGEL, 1999], [BLACK, 1995], [3], [10].

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**H**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
A	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
B	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
C	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
D	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
E	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
F	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
G	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
H	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
I	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
J	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
K	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
L	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
M	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
N	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
O	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
P	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Q	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
R	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
S	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
T	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
U	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
V	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
W	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
X	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
Y	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
Z	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Tabla 2.6 Tabla de *Vigenère*

La clave del sistema de cifrado de *Vigenère* es una palabra de k letras, $k \geq 1$ siempre, del alfabeto Z^{26} utilizado anteriormente. Esta palabra es un elemento del producto cartesiano Z^{26}x, Z^{26}x. . .xZ^{26} (k veces), que es justamente el alfabeto del cifrado de *Vigenère*. De esta forma, el mensaje a cifrar en texto claro ha de descomponerse en bloques de k elementos (letras) y aplicar sucesivamente la clave empleada a cada uno de estos bloques, utilizando la tabla anteriormente proporcionada

Para cifrar se localiza la letra del texto claro en la primera fila y se busca la letra de la clave en la primera columna, y la intersección de las dos coordenadas proporcionara la letra cifrada. Para descifrar se procede de forma inversa, se busca la letra clave en la primera columna y se localiza en esa fila la letra cifrada, una vez localizada se sube por esa columna hasta la primera fila que dará la letra en claro [ÁNGEL, 1999], [2], [3], [10].

Ejemplo:

Se quiere codificar la frase **La abrumadora soledad del programador** utilizando la clave **prueba**. En primer lugar, de debe tomar en cuenta la longitud de la clave: es de seis caracteres. Así, descompone la frase en bloques de longitud seis; aunque el último bloque es de longitud tres, esto no afecta para nada al proceso de cifrado:

laabru madora soleda ddelpr ograma dor

Ahora, se aplica a cada bloque la clave prueba y se busca los resultados como entradas de la tabla de *Vigenère*:

Texto claro:	**laabru**	**madora**	**soleda**	**ddelpr**	**ograma**	**dor**
Clave:	**prueba**	**prueba**	**prueba**	**prueba**	**prueba**	**pru**
Texto cifrado:	**arufsu**	**brxhsa**	**igfiea**	**suyoqr**	**exmena**	**sgm**

Este método de cifrado poli alfabético se consideraba invulnerable hasta que en el siglo XIX se consiguieron descifrar algunos mensajes codificados con este sistema, mediante el estudio de la repetición de bloques de letras: la distancia entre un bloque y su repetición suele ser múltiplo de la palabra tomada como clave.

Una mejora sobre el cifrado de *Vigenère* fue introducida por el sistema de *Vernam*, utilizando una clave aleatoria de longitud k igual a la del mensaje. La confianza en este nuevo criptosistema hizo que se utilizara en las comunicaciones confidenciales entre la Casa Blanca y el Kremlin, hasta, por lo menos, el año 1987 [10].

2.1.1.4 MATRIZ DE *PLAYFAIR*

El cifrado de *Playfair* en realidad fue inventado por *Charles Wheatstone*, para comunicaciones telegráficas secretas en 1854, no obstante se le atribuye a su amigo el científico *Lyon Playfair*.

Utilizado por el Reino Unido en la Primera Guerra Mundial, este sistema consiste en separar el texto en claro en diagramas y proceder a su cifrado de acuerdo a una matriz alfabética de dimensiones 5 X 5 en la cual se encuentran representadas las 26 letras del alfabeto ingles como se muestra en la tabla 2.7, y para una mayor seguridad se puede agregar una palabra clave [BLACK, 1995], [2].

A	**B**	**C**	**D**	**E**
F	**G**	**H**	**I/J**	**K**
L	**M**	**N**	**O**	**P**
Q	**R**	**S**	**T**	**U**
V	**W**	**X**	**Y**	**Z**

Tabla 2.7 Matriz Alfabética

Añadiendo una palabra clave a la matriz de cifrado se obtiene una mayor seguridad, la clave se sitúa al comienzo de la matriz quitando las repeticiones y a continuación el resto de las letras del alfabeto.

Para cifrar es necesario seguir las siguientes reglas [BLACK, 1995], [2]:

1. Si los símbolos están en la misma fila y diferente columna, se desplaza una columna a la derecha.

$$(a_{ij}; a_{ik}) \longrightarrow (a_{ij+1}; a_{ik+1})$$

2. Si los símbolos están en la misma columna y diferente fila, se desplaza una columna hacia abajo.

$$(a_{ik}; a_{jk}) \longrightarrow (a_{(i+1)k}; a_{(j+1)k})$$

3. Si están en filas y columnas diferentes.

$$(a_{ki}; b_{js}) \longrightarrow (a_{ks}; b_{ji})$$

4. Si hay dos símbolos iguales consecutivos, se inserta un símbolo acordado con anterioridad (por lo general es "X").

Ejemplo:

Palabra Clave: **VERANO AZUL**

Letra Nula: **X**

V	E	R	A	N
O	Z	U	L	B
C	D	F	G	H
I/J	K	M	P	Q
S	T	W	X	Y

Tabla 2.8 Matriz Alfabética con Clave

Texto claro:	**COMPRUEBALO TÚ**
Con la letra nula:	**CO MP RU EB AL OT UX**
Texto cifrado:	**IC PQ UF NZ LG ZS LW**

2.1.2 ALGORITMOS DE TRASPOSICION

Los algoritmos de sustitución y los códigos, mantienen el orden de los símbolos en claro, pero los disfrazan. En comparación con éstos, los algoritmos de transposición reordenan las letras pero no las disfrazan.

El algoritmo de transposición más común es el de tipo columnar; la clave del cifrador debe ser una palabra que no tenga ninguna letra repetida, en el ejemplo que se presenta a continuación la clave es la palabra MEGABUCK. El propósito de la clave es de numerar las diferentes columnas que se formarán, de forma que la columna 1 es aquella que queda bajo la letra de la clave más próxima al principio del alfabeto y así sucesivamente. El texto en claro se escribe debajo de la clave en renglones horizontales; el texto cifrado se lee por columnas, comenzando por la columna cuya letra clave tiene el menor valor como se observa en la tabla 2.9 [BLACK, 1995], [2], [3].

Texto en claro:

Por favor transfiera un millón de dólares

Clave de cifrado: **M E G A B U K**

M	E	G	A	B	U	K
6	3	4	1	2	7	5
p	o	r	f	a	v	o
r	t	r	a	n	s	f
i	e	r	a	u	n	m
i	l	l	o	n	d	e
d	o	l	a	r	e	s

Tabla 2.9 Tabla de Cifrado

Texto cifrado:

faaoa	anunr	otelo	rrrll	ofmes	priid

Tabla 2.10 Tabla de Texto Cifrado

2.2 CRIPTOGRAFÍA DE CLAVE PRIVADA

De acuerdo a infinidad de autores la encriptación convencional (también conocida como Simétrica) o los criptosistemas de clave privada están basados en una privacidad donde la clave secreta que es idéntica para el emisor tanto como para el receptor. El emisor del mensaje genera una clave y posteriormente le transmite en un canal seguro al receptor o receptores autorizados a recibir los mensajes como puede observar en la figura 2.1. Aunque la distribución de claves en un gran problema para los sistemas simétricos. Ya que se utiliza la misma clave para encriptar y desencriptar los datos, esta clave se debe mantener en privado [ÁNGEL, 1999], [FROUFE, 1997], [OPPLIEGER, 1988].

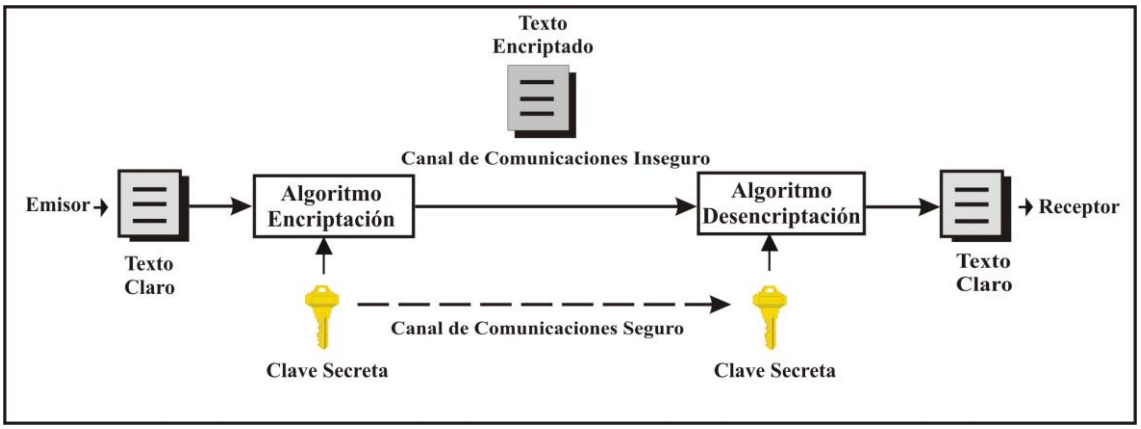

Figura 2.1 Criptografía Simétrica

Estos sistemas sólo permiten confidencialidad y no autenticación ni firma digital.

De acuerdo a la encriptación simétrica se refiere a algoritmos de encriptación en donde el algoritmo de encriptación y desencriptación emplean la misma clave. Específicamente [TANENBAUN, 1997a]:

$$E(p, k)=C \ \& \ D(C, k)=p$$

Donde:

E = algoritmo de encriptación
D = algoritmo de decriptación
P = texto plano o plaintext (datos originales)
K = clave de encriptación
C = código de texto o ciphertext

Las desventajas de este tipo de sistemas son principalmente:

1. La dificultad para transmitir la clave a utilizar por cada pareja.
2. El gran número de claves que se requieren cuando existen un gran número de usuarios.

La criptografía con clave secreta se lleva utilizando desde hace millones de años, ésta ha podido ser implementada en diferente dispositivos, manuales, mecánicos, eléctricos, hasta los algoritmos actuales que son programables en cualquier computadora. Los principales algoritmos simétricos actuales son DES, IDEA, RC5 y el novedoso AES [OPPLIEGER, 1988], [FROUFE, 1997].

Debido a la gran cantidad de algoritmos creados y utilizados solo se describirán los mencionados anteriormente y se profundizara en el algoritmo de cifrado DES y sus variantes debido a que es el algoritmo más utilizado por el servidor sistema de autentificación **Kerberos**.

2.2.1 DES (Data Encryption Standard)

En 1971 IBM inventó un algoritmo de encriptación simétrico basado en la aplicación de todas las teorías existentes sobre criptografía. Se llamó LUCIFER y funcionaba con claves simétricas de 128 bits. Fue vendido en exclusividad a la empresa de seguros *Lloyd's*.

En 1973, durante la administración de *Richard Nixon*, la oficina nacional de estándares (*National Bureau of Standard*) publicó una solicitud de propuestas de algoritmo criptográficos para proteger la transmisión y almacenamiento de datos. Y fue hasta 1974 cuando IBM entregó su algoritmo candidato “LICIFER” pero debido a que utilizaba una clave de 112 bits fue reducido a 56 por la NSA (Agencia Nacional de Seguridad) y esto originó controversia debido a que se pensó que la NSA había debilitado intencionadamente el algoritmo del DES para desencriptarlo. Fue evaluado por la NBS y adoptado con una modificación como el nuevo Estándar de Encriptamiento de Datos en 1977 y es reafirmado cada cinco años. Todas las agencias federales de Estados Unidos y otras organizaciones que procesan información, utilizan DES (para documentos no clasificados), su uso es también común entre corporaciones no gubernamentales [TANENBAUN, 1997b].

DES utiliza una clave de 64 bits; sin embargo, 8 bits son para detección de errores, por ello DES un sistema de 56 bits para propósitos de seguridad. Se le representa como un código de bloque ya que encripta datos en bloques de 64 bits de datos binarios. La seguridad de DES está basada en el secreto de la clave, no en el secreto del algoritmo.

La seguridad se aumenta más por el tamaño de la clave ya que hay setenta mil billones (70, 000, 000, 000, 000, 000) de claves posibles; de ese modo, las oportunidades de obtener la clave son lo suficientemente bajas como para que sean seguras para la mayoría de los ambientes distribuidos. Por supuesto que con la capacidad de la PC promedio que se incrementa perennemente, la habilidad para buscar consecutivamente una clave y romper un código proporcionalmente está incrementándose también [ÁNGEL, 1999], [CABALLERO, 1996], [TANENBAUN,1997a], [TANENBAUN,1997b].

El algoritmo DES cifra la información por bloques, es decir, el texto en claro se descompone en bloques de 64 bits que serán encriptados uno tras otro. Se utiliza el mismo algoritmo para encriptar como para desencriptar, aunque no se utiliza de la misma forma en ambos casos.

DES utiliza claves de 56 bits que son los significativos, aunque estas suelen distribuirse en forma de números de 64 bits. De estos 64 bits, uno de cada ocho,

es utilizado como bit de paridad (64-8=56). Una clave de 56 bits hace que el conjunto de claves utilizables se reduzca a $2^{56} = 7.2 \text{ x } 10^{16}$ posibles claves o llaves diferentes.

Como se observa en la figura 2.3 (a), el texto normal se cifra en bloques de 64 bits, produciendo 64 bits de texto cifrado. El algoritmo, que se define con la clave de 56 bits, tiene 19 etapas diferentes. La primera etapa es una transposición, independiente de la clave, del texto normal de 64 bits. La última etapa es el inverso exacto de esta transposición. La etapa previa a la última intercambia los 32 bits de la izquierda y los 32 bits de la derecha. Las 16 etapas (también conocidas como Iteraciones o vueltas) restantes son funcionalmente idénticas como se observa en la figura 2.3 (b), pero se definen mediante diferentes funciones de la clave. El algoritmo se ha diseñado para permitir que el descifrado se haga con la misma clave que el cifrado, simplemente ejecutando los pasos en orden inverso.

Cada una de las 16 etapas intermedias toma dos entradas de 32 bits cada una: L_i (mitad izquierda) y R_i (mitad derecha). Los subíndices i indican lo subíndices que son incrementados en cada una de las 16 iteraciones y produce dos salidas de 32 bits. La salida de la izquierda es simplemente una copia de la entrada de la derecha. La salida de la derecha es el or exclusivo (XOR = $\oplus$) a nivel de bit de la entrada izquierda y una función de la entrada derecha y la clave de esta etapa K_i. Toda la complejidad reside en esta función, como se puede observar en la figura 2.3 (b) del el donde se detalla el proceso de una iteración.

La función consiste en cuatro pasos, ejecutados en secuencia. Primero se construye un número de 48 bits, E, expandiendo el R_{i-1} de 32 bits según una regla fija de transposición y duplicación. Después, se aplica un or exclusivo (XOR = $\oplus$) a E y K_i. Esta salida entonces se divide en ocho grupos de 6 bits, cada uno de los cuales se alimenta a una **caja S** distinta. Cada una de las 64 entradas posibles a la **caja S** se transforma en una salida de 4 bits. Por último estos 8 x 4 bits se pasan a través de una **caja P** [CABALLERO, 1996] [HALSALL, 1998], [TANENBAUN, 1997a].

Las trasposiciones y las sustituciones que realizan las **cajas P** y **cajas S** se implementan mediante circuitos sencillos como se puede observar en la figura 2.2(a) se tiene el dispositivo conocido como **cajas P** (la letra P significa permutación), esta se utiliza para realizar una transposición de una entrada de 8 bits de arriba hacia abajo como 01234567 y la salida de dicha caja es 36071245. Las sustituciones se llevan acabo mediante las **cajas S**, como se muestra en la figura 2.2 (b). En dicho ejemplo de la **caja S**, se ingresa un texto normal de 3 bits y sale un nuevo texto cifrado de 3 bits, la entrada de 3 bits selecciona una de las ocho salidas de la primera etapa y la establece en 1; las demás líneas son 0.La segunda etapa es una **caja P**. Y la tercera etapa codifica en binario nuevamente la línea de entrada de selección. Con el alambrado que se muestra, si los ocho números octales 01234567 entraran uno tras otro, la secuencia de salida seria de 24506713.

La potencia real de estos elementos básicos solo se hacen aparentes cuando colocan en cascada una serie completa de cajas para formar un cifrado de producto como se muestra en la figura 2.2(c) [HALSALL, 1998], [TANENBAUN, 1997b.]

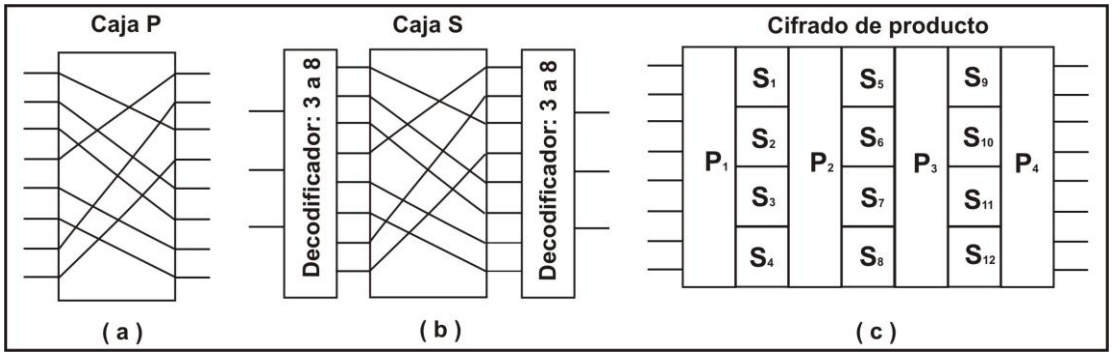

Figura 2.2 Elementos Básicos de Cifrado de Producto. (a) Cajas P. (b) Caja S. (c) Producto.

En cada una de las 16 iteraciones, se usa una clave diferente. Antes de iniciarse el algoritmo, se aplica una transposición de 56 bits a la clave. Justo antes de cada iteración, la clave se divide en dos unidades de 28 bits, cada una de las cuales se gira hacia la izquierda una cantidad de bits dependiente del número de iteración.

K_i se deriva de esta clave girada aplicándole otra transposición de 56 bits. En cada vuelta (o iteración) se extrae y permuta de los 56 bits un subgrupo de 48 bits diferente [CABALLERO, 1996], [HALSALL, 1998], [TANENBAUN, 1997a].

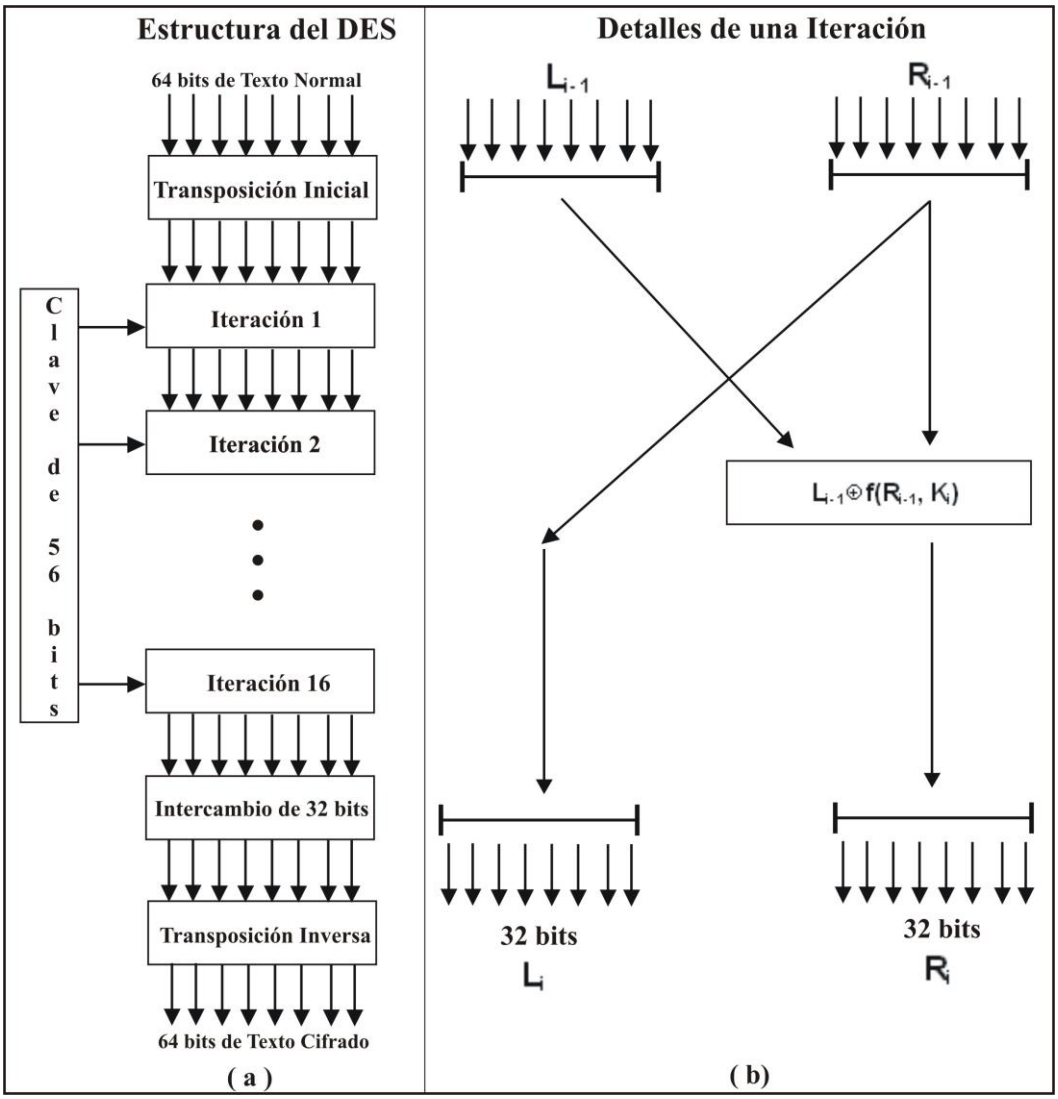

Figura 2.3 Estructura estándar del DES. (a) Esbozo general. (b) Esquema de una iteración.

Existen cuatro modos de funcionamiento de este algoritmo, dos de las cuales además de cifrar comprueban que el mensaje recibido concuerda con el enviado, verificando la precisión de la transmisión y detectando posibles ruidos o problemas en la comunicación.

Aunque ha sido ampliamente usado, estimándose en varios cientos de miles los dispositivos físicos que lo implementan vendidos en el mundo, siempre existieron dudas acerca de su seguridad, fundamentalmente por el tamaño excesivamente reducido de su clave. Se escribieron artículos detallando cómo se podrían construir dispositivos de propósito específico capaces de romper el código en pocos días. También se hicieron pruebas en las que mediante la división y el reparto de su espacio de claves entre miles de PC's de usuarios de Internet, consiguieron romperlo en semanas.

Finalmente se construyó una máquina con el propósito específico de reventar mensajes cifrados en DES en una media de 35 horas. Dado que la máquina descubre la clave de cifrado mediante una búsqueda exhaustiva, es de prever que DES deje de utilizarse en un lapso de tiempo relativamente corto. Se puede decir que DES ha sido víctima de su propia eficiencia, ya que las enormes velocidades de cifrado y descifrado conseguidas han sido las que a la postre han posibilitado certificar su desaparición [FROUFE, 1997], [TANENBAUN, 1997b].

Los principales inconvenientes que presenta DES son [FROUFE, 1997]:

- Se considera un secreto nacional de EEUU, por lo que está protegido por leyes determinadas y no se puede comercializar ni en hardware ni en software fuera de ese país sin permiso del Departamento de Estado.
- La clave es corta, tanto que no asegura una fortaleza adecuada. Hasta ahora había resultado suficiente, y nunca había sido roto el sistema. Pero con la potencia de cálculo actual y futuro de las computadoras y con el trabajo en equipo por Internet se creé que se puede violar el algoritmo.
- No admite longitud de clave variable, con lo que sus posibilidades de configuración son muy limitadas.
- La seguridad del sistema se ve comprometida ampliamente si se conoce un número suficiente textos elegidos, ya que existe un sistema matemático, llamado Criptoanálisis Diferencial.

Entre sus ventajas cabe citar [FROUFE, 1997]:

- Es el sistema más usado del mundo, el más barato y el más probado.
- Es muy rápido y fácil de implementar.
- Desde su aparición nunca ha sido roto con un sistema práctico.

2.2.1.1 TRIPLE DES

En 1979, IBM se dio cuenta de que la longitud de la clave del DES era demasiado corta, así que diseño una forma de aumentarla efectivamente utilizando codificación triple, dicho método está basado en tres iteraciones sucesivas del algoritmo DES, con lo que se consigue una longitud de clave de 128 bits, y que es compatible con DES simple.

Por lo tanto, el cifrado con el algoritmo DES no es tan seguro, sin embargo, el cifrado triple con el algoritmo DES aumenta la seguridad. El método seleccionado, que se ha incorporado al estándar internacional 8732, se ilustra en la siguiente figura 2.4 donde se utiliza la letra **P** para representar el texto normal, la letra **K** para la clave y por último la letra **C** para el texto cifrado.

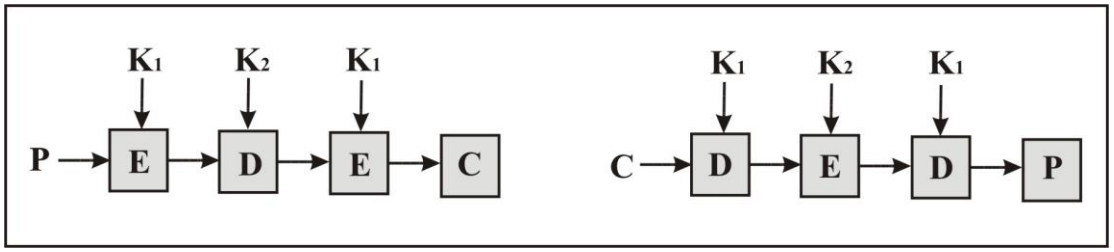

Figura 2.4 Esquema del Triple DES

Aquí se usan dos claves (**K_1** y **K_2**) y tres etapas. En la primera etapa el texto normal se cifra con K_1. En la segunda etapa el algoritmo DES se ejecuta en modo de descifrado, usando K_2 como clave. Por último, se hace otro cifrado usando K_1 [TANENBAUN, 1997b], [7].

El hecho de que se usen dos claves y en modo EDE (cifrado-descifrado-cifrado) en lugar de EEE (cifrado-cifrado-cifrado) es debido a dos motivos [TANENBAUN, 1997b]:

1. En primer lugar, los criptógrafos admiten que 112 bits son suficientes para la encriptación de la información. Subir a 168 bits (3 claves) simplemente agregaría carga extra innecesaria de administrar y transportar otra clave.

2. En segundo lugar, la razón de cifrar, descifrar y luego cifrar de nuevo es la compatibilidad con los sistemas DES de una sola clave. Tanto las funciones de cifrado como de descifrado son correspondencias entre números de 64 bits. Desde el punto de vista criptográfico, las dos correspondencias son igualmente robustas. Sin embargo, usando EDE en lugar de EEE, una computadora que usa cifrado triple puede hablar con otra que usa cifrado sencillo simplemente estableciendo $K_1 = K_2$. Esta propiedad permite la introducción gradual del cifrado triple.

2.2.2 IDEA (International Data Encryption Algorithm)

Después de comprobar la debilidad del algoritmo DES en su forma simple, diversos trabajos propusieron nuevos métodos de cifrados de bloques Sin embargo el más interesante e importante de los cifrados posteriores al algoritmo DES es el algoritmo IDEA.

El algoritmo IDEA (Algoritmo de Encriptación de Datos Internacionales) es bastante más joven que DES, pues fue creado en 1992 por *Lai* y *Massey* en *ETH Zurich* en Suiza. Para muchos constituye el mejor y más seguro algoritmo simétrico disponible en la actualidad. Trabaja con bloques de 64 bits de longitud y emplea una clave de 128 bits. Tal que en el caso de DES, se usa el mismo algoritmo tanto para cifrar como para descifrar.

La estructura básica del algoritmo se asemeja al algoritmo DES en cuanto a que se alteran bloques de entrada de texto normal de 64 bits en una secuencia de iteraciones definidas para producir bloques de salida de texto cifrado de 64 bits, como se puede ver en la figura 2.5(a) [TANENBAUN,1997b].

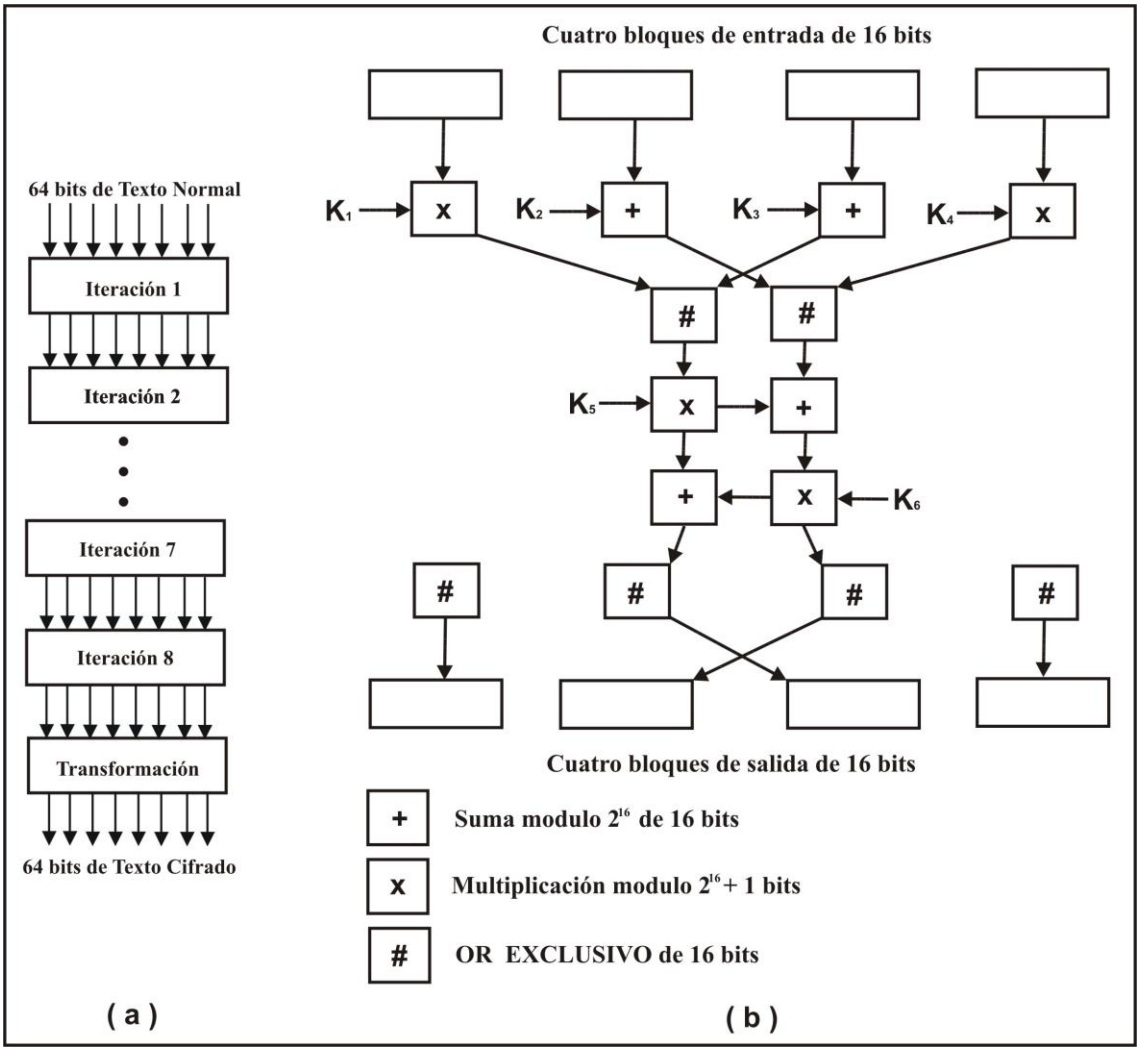

Figura 2.5 (a) Esquema General de IDEA. (b) Detalle de una Iteración.

Dada la amplia alteración de bits (por cada iteración, cada uno de los bits de salida depende de cada uno de los bits de entrada), basta con ocho iteraciones. Como con todos los cifrados de bloque, el algoritmo IDEA también puede usarse en el modo de realimentación de cifrado y en los demás modos del algoritmo DES.

Los Detalles de cada iteración se puede ver en la figura 2.5 (b) El algoritmo IDEA usa tres operaciones, todas sobre números sin signo de 16 bits.

Dichas operaciones son las siguientes:

- or exclusivo (XOR = $\oplus$)
- suma módulo 2^{16}
- multiplicación módulo $2^{16}+1$

Las tres operaciones se pueden efectuar fácilmente en una microcomputadora de 16 bits ignorando las partes de orden mayor de los resultados. Las operaciones tienen la propiedad de que ningunos dos pares obedecen la ley asociativa ni la ley distributiva, dificultando el criptoanálisis. La clave de 128 bits se usa para generar 52 subclaves de 16 bits cada una, 6 por cada una de las ocho iteraciones y 4 para la transformación final. El descifrado usa el mismo algoritmo que el cifrado, sólo que con subclaves diferentes [ÁNGEL, 1999], [FROUFE, 1997], [HALSALL, 1998], [TANENBAUN, 1997b], [2].

El algoritmo de desencriptación es muy parecido al de encriptación, por lo que resulta muy fácil y rápido de programar, y hasta ahora no ha sido roto nunca, aportando su longitud de clave una seguridad fuerte ante los ataques por fuerza bruta (prueba y ensayo o diccionarios). Este algoritmo es de libre difusión y no está sometido a ningún tipo de restricciones o permisos nacionales, por lo que se ha difundido ampliamente, utilizándose en sistemas como *UNÍX* [FROUFE, 1997], [2].

2.2.3 RC5

El sistema criptográfico simétrico RC5 es el sucesor de RC4, frente al que presenta varias mejoras. RC4 consiste en realizar un XOR al mensaje con un arreglo que se supone aleatorio y que se desprende de la clave, mientras que RC5 usa otra operación, llamada dependencia de datos, que aplica a los datos para

obtener así el mensaje cifrado. Ambos han sido diseñados por RSA *Data Security Inc.*, es actualmente una de las más importantes en el campo de los sistemas de cifrado y protección de datos. Permite diferentes longitudes de clave (aunque está prohibida su exportación fuera de EEUU con longitudes superiores a 56 bits), y funciona como un generador de números aleatorios que se suman al texto mediante una operación de tipo OR-Exclusiva. Es además ampliamente configurable, permitiendo fijar diferentes longitudes de clave, número de iteraciones y tamaño de los bloques a cifrar, por lo que le permite adaptarse a cualquier aplicación. Por ejemplo, este algoritmo es el usado por *Nestcape* para implementar su sistema de seguridad en comunicaciones SSL (*Secure Socket Layer*) [ÁNGEL, 1999], [FROUFE, 1997], [2].

En cuanto a su seguridad, aún es pronto para afirmar nada concluyente, aunque en 1996 una universidad francesa consiguió romper el sistema RC4 con clave de 40 bits, lo que hace sospechar que RC5 con longitudes de clave de 56 bits no es lo suficientemente seguro [FROUFE, 1997].

2.2.4 AES *(Advanced Encryption Standard)*

El NIST (*National Institute for Standards and Technology*) de EEUU, en busca de un nuevo sistema de encriptación simétrico que reuniera las características funcionales y de seguridad necesarias, decidió convocar en 1977 un concurso a nivel mundial, invitando a los principales desarrolladores de este tipo de sistemas a crear un algoritmo que pudiera ser tomado como estándar para los siguientes años. Es considerado el sucesor de DES. Este algoritmo se adoptó oficialmente en octubre del 2000 como nuevo Estándar Avanzado de Cifrado (AES) por el NIST para su empleo en aplicaciones criptográficas.

Sus autores de dicho algoritmo son dos, los belgas *Joan Daemen* y *Vincent Rijmen*, de ahí su nombre con el cual también se le conoce *"Rijndael"*. Tiene como peculiaridad que todo el proceso de selección, revisión y estudio tanto de este

algoritmo como de los restantes candidatos, se efectuó de forma pública y abierta, por lo que, toda la comunidad criptográfica mundial ha participado en su análisis, lo cual convierte a AES en un algoritmo perfectamente digno de la confianza de todos [ÁNGEL, 1999], [CABALLERO, 1996], [FROUFE, 1997], [HALSALL, 1998], [TANENBAUN, 1997b].

AES es un sistema de cifrado por bloques, diseñado para manejar longitudes de clave y de bloque variables, ambas comprendidas entre los 128 y los 256 bits. Es un algoritmo que se basa en aplicar un número determinado de rondas a un valor intermedio que se denomina estado. Dicho estado puede representarse mediante una matriz rectangular de bytes, que posee cuatro filas, y Nb columnas. Así, por ejemplo, si nuestro bloque tiene 160 bits como se ve en la tabla 2.11, Nb será igual a 5.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$	$a_{0,4}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,4}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$

Tabla 2.11 Matriz Rectangular de Bytes

La clave tiene una estructura análoga a la del estado, y se representará mediante una tabla con cuatro filas y N_k columnas. Si nuestra clave tiene, por ejemplo, 128 bits, N_k será igual a 4, en la siguiente tabla 2.12

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Tabla 2.12 Representación de una Estructura Análoga

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

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

1. Calcular K_0, K_1, …K_n subclaves a partir de la clave K.
2. $S \leftarrow B \oplus K_0$
3. Para i = 1 hasta n hacer
4. Aplicar ronda i-ésima del algoritmo con la subclave

El algoritmo de descifrado consistirá en aplicar las inversas de cada una de las funciones en el orden contrario, y utilizar los mismos Ki que en el cifrado, sólo que comenzando por el último.

Es un algoritmo resistente al criptoanálisis tanto lineal como diferencial y uno de los más seguros en la actualidad ya que para recuperar la clave a partir de un par texto cifrado-texto claro hay que realizar una búsqueda exhaustiva.

Debido a que AES permite emplear diferentes longitudes tanto de bloque como de clave, el número de rondas requerido en cada caso es variable. En la tabla 2.13, se especifica cuantas rondas son necesarias en función de los tamaños de clave y bloque, Nb y Nk.

	N_b=4 (128 bits)	N_b=6 (192 bits)	N_b=8 (256 bits)
N_k=4 (128 bits)	10	12	14
N_k=6 (192 bits)	12	12	14
N_k=8 (256 bits)	14	14	14

Tabla 2.13 Especificación de Rondas

Siendo S la matriz de estado, y K_i la subclave correspondiente a la ronda i-ésima, cada una de las rondas posee la siguiente estructura:

1. S ← ByteSub(S)
2. S ← DesplazarFila(S)
3. S ← MezclarColumnas(S)
4. S ← Ki ⊕ S

Estas cuatro funciones están diseñadas para proporcionar resistencia frente a criptoanálisis lineal y diferencial. Cada una de las funciones tiene un propósito:

- Funciones DesplazarFila y MezclarColumnas permiten obtener un alto nivel de difusión a lo largo de varias rondas.
- Función ByteSub consiste en la aplicación paralela de s-cajas.
- La capa de adición de clave es un simple or-exclusivo entre el estado intermedio y la subclave correspondiente a cada ronda.

La última ronda es igual a las anteriores, pero eliminando el paso 3 [ÁNGEL, 1999], [HALSALL, 1998], [TANENBAUN, 1997a], [TANENBAUN, 1997b].

2.3 CRIPTOGRAFÍA CON CLAVE PÚBLICA

En 1976 *Whitfield Diffie* y *Martin Hellman* de la Universidad de *Stanford* publicaron el artículo "***New directions in cryptography***". En él proponían un nuevo tipo de criptografía basado en utilizar claves distintas para encriptar y desencriptar, una de ellas se hace pública y la otra es privada de cada usuario. Así todos los usuarios de la red tienen acceso a las claves públicas, pero únicamente ellos conocen su clave privada. Estas ideas supusieron la revolución de la criptografía: se podía utilizar para confidencialidad (como los sistemas simétricos), autenticación y firma digital, además de solucionar el problema de la distribución de claves simétricas. Pero no fue hasta la invención del RSA de *Rivest Shamir* y *Adleman* publicado en

1978 que hoy en día es uno de los más usados, cuando toma forma la criptografía asimétrica [ÁNGEL, 1999], [TANENBAUN, 1997b], [7], [11].

Este se basa en el uso de dos claves diferentes, claves que tienen una propiedad fundamental: una clave puede desencriptar lo que la otra ha encriptado. Generalmente una de las claves de la pareja, denominada **clave privada**, es usada por el propietario para encriptar los mensajes, mientras que la otra, llamada **clave pública**, es usada para desencriptar el mensaje cifrado.

Las claves pública y privada tienen propiedades matemáticas especiales, de tal forma que se generan siempre a la vez, por parejas, estando cada una de ellas ligada intrínsecamente a la otra, de tal forma que si dos llaves públicas son diferentes, entonces sus llaves privadas asociadas también lo son, y viceversa.
Los algoritmos asimétricos están basados en funciones matemáticas fáciles de resolver en un sentido, pero muy complicadas de realizar en sentido inverso, salvo que se conozca la clave privada, como la potencia y el logaritmo. Ambas claves, pública y privada, están relacionadas matemáticamente, pero esta relación debe ser lo suficientemente compleja como para que resulte muy difícil obtener una a partir de la otra. Este es el motivo por el que normalmente estas claves no las elige el usuario, si no que lo hace un algoritmo específico para ello, y suelen ser de gran longitud.

Mientras que la clave privada debe mantenerla en secreto su propietario, ya que es la base de la seguridad del sistema, la clave pública es difundida ampliamente por Internet, para que esté al alcance del mayor número posible de personas, existiendo servidores que guardan, administran y difunden dichas claves.

En este sistema, para enviar un documento con seguridad, como se puede observar en figura 2.6, el emisor encripta el mismo con la clave pública del receptor y lo envía por el medio inseguro. Este documento está totalmente protegido en su viaje, ya que sólo se puede desencriptar con la clave privada

correspondiente, conocida solamente por el receptor. Al llegar el mensaje cifrado a su destino, el receptor usa su clave privada para obtener el mensaje en claro [ÁNGEL, 1999], [TANENBAUN, 1997a], [TANENBAUN, 1997b].

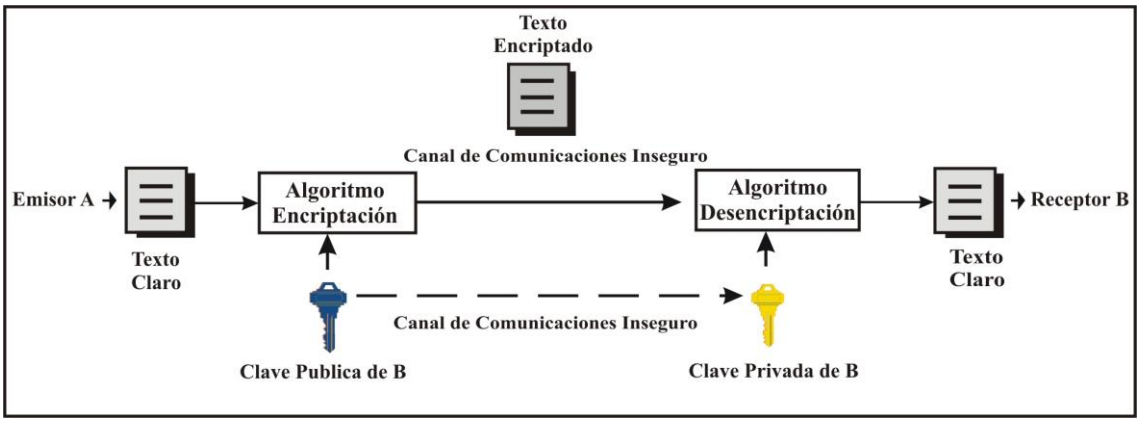

Figura 2.6 Encriptación con Clave Pública

Una variación de este sistema se produce cuando es el emisor A, quien encripta un texto con su clave privada, enviando por el medio inseguro tanto el mensaje en claro como el cifrado. Así, cualquier receptor B del mismo puede comprobar que el emisor ha sido A, y no otro que lo suplante, con tan sólo desencriptar el texto cifrado con la clave pública de A y comprobar que coincide con el texto sin cifrar. Como sólo A conoce su clave privada, B puede estar seguro de la autenticidad del emisor del mensaje. Este sistema de autentificación de denomina **firma digital,** que se explicara más adelante [7].

En teoría, la criptografía con clave pública puede ser más conveniente que la criptografía con clave secreta, ya que evita que las dos entidades (emisor y receptor) tengan que compartir la misma llave. Sin embargo este tipo de sistemas requieren generalmente operaciones aritméticas difíciles de realizar para pequeños procesadores así como el tiempo en el que realizan dichas operaciones [OPPLIEGER, 1998].

Dentro de los principales algoritmos de encriptación con clave pública actuales se encuentran *Deffie-Hellman*, RSA, Curvas Elípticas y DSS.

2.3.1 DEFFIE-HELLMAN

Este algoritmo de encriptación de *Whitfield Diffie* y *Martin Hellman* supuso una verdadera revolución en el campo de la criptografía, ya que fue el punto de partida para los sistemas asimétricos, basados en dos claves diferentes, la pública y la privada. Publicado en 1976, surgiendo como ilustración del artículo "*New directions in Cryptography*". Su importancia se debe sobre todo al hecho de ser el inicio de los sistemas asimétricos, ya que en la práctica sólo es válido para el intercambio de claves simétricas, y con esta funcionalidad es muy usado en los diferentes sistemas seguros implementados en Internet, como SSL (*Secure Socket Layer*) y VPN (*Virtual Private Network*) [ÁNGEL,1999], [7].

Matemáticamente se basa en las potencias de los números y en la función mod (módulo discreto). Uniendo estos dos conceptos se define la potencia discreta de un número como $\mathbf{Y = X^a \bmod q}$. Si bien el cálculo de potencias discretas es fácil, la obtención de su función inversa, el logaritmo discreto, no tiene una solución analítica para números grandes.

Para implementar el sistema re realizan los siguientes pasos [Ángel, 1999], [7]:

1. Se busca un número primo muy grande, **q**.

2. Se obtiene el número **ß**, raíz primitiva de **q**, es decir, que cumple que $\mathbf{ß \bmod q}$, , $\mathbf{ß^2 \bmod q}$,...., $\mathbf{ß^{q-1}}$ mod q son números diferentes.

3. **ß** y **q** son las claves públicas.

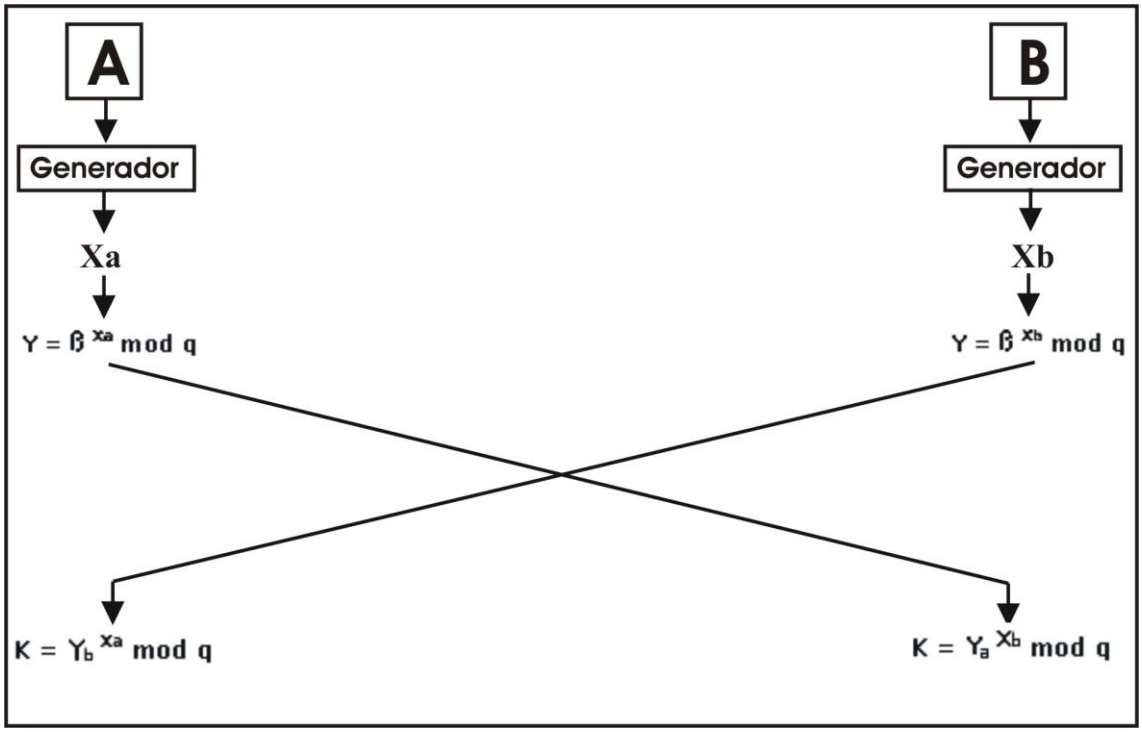

Figura 2.7 Esquema de la Transmisión de la Clave con *Deffie-Hellman*

Como se ve en la figura 2.7 para generar una clave simétrica compartida entre dos usuarios, **A** y **B**, ambos parten de un generador de números seudo aleatorios, que suministra un número de este tipo diferente a cada uno, $\mathbf{X_a}$ y $\mathbf{X_b}$. Estos son las claves privadas de **A** y **B**. Con estos números y las claves públicas **ß** y **q** que ambos conocen, cada uno genera un número intermedio, $\mathbf{Y_a}$ e $\mathbf{Y_b}$, mediante las fórmulas:

$$\mathbf{Y_a = ß^{Xa} \bmod q}$$
$$\mathbf{Y_b = ß^{Xb} \bmod q}$$

Estos números son intercambiados entre ambos, posteriormente cada uno opera con el que recibe del otro, obteniendo en el proceso el mismo número ambos:

$$\mathbf{K = Y_b{}^{Xa} \bmod q}$$
$$\mathbf{K = Y_a{}^{Xb} \bmod q}$$

Este número K es la clave simétrica que a partir de ese momento ambos comparten, y que pueden usar para establecer una comunicación cifrada mediante cualquiera de los sistemas simétricos. Con este esquema, si se desea compartir una clave privada con otro usuario cualquiera, basta con acceder a su Y_u y enviarle la nuestra. Para facilitar este proceso se suelen publicar las Y_u de todos los usuarios interesados en un directorio de acceso común) [ÁNGEL, 1999], [TANENBAUN, 1997b], [7].

2.3.2 RSA

Es el más popular y utilizado de los algoritmos asimétricos. Fue inventado en 1978 por *Rivest*, *Shamir* y *Adleman* que dan nombre al algoritmo. Patentaron el algoritmo y cuando alcanzó popularidad fundaron una empresa, *RSA Data Security Inc.*, para la explotación comercial. Para su implementación y comercialización se deben pagar derechos a esta empresa, no obstante actualmente se encuentran muchas versiones gratuitas en Internet [TANENBAUN, 1997b].

El algoritmo utiliza las siguientes claves:

- Como clave públicas dos números grandes elegidos por un programa: **e** y **n**.
- Como clave privada un número grande **d**, consecuencia de los anteriores.

El cálculo de estas claves se realiza en secreto en la máquina encargada de la clave privada. Este proceso tiene gran importancia para la posterior seguridad del sistema.

El proceso que se realiza en este algoritmo es el siguiente:

1. Se buscan dos números grandes (entre 100 y 300 dígitos) y primos: **p** y **q,** de manera que **p ≠ q** .
2. Se calcula ϕ **= (p-1) * (q-1)** y **n = p** * **q**.
3. Se busca **e** como un número sin múltiplos comunes a ϕ.
4. Se calcula $\mathbf{d = e^{-1} \bmod \phi}$. (mod = resto de la división de enteros).
5. Se hacen **públicas** las claves **n** y **e**, se guarda **d** como clave privada y se destruyen **p**, **q** y ϕ.

Aquí se muestra un ejemplo de cifrado/descifrado con RSA. Los parámetros usados aquí son pequeños y orientativos con respecto a los que maneja el algoritmo:

p = 61 1º nº primo Privado
q = 53 2º nº primo Privado
n = 3233 Producto p*q
e = 17 Exponente Público
d = 2753 Exponente Privado

- La clave pública (e, n). La clave privada es d. La función de cifrado es:

$$encrypt\,(m) = m^e \pmod n = m^{17} \pmod{3233}$$

- Donde m es el texto sin cifrar La función de descifrado es:

$$decrypt(c) = c^d \pmod n = c^{2753} \pmod{3233}$$

- Donde c es el texto cifrado. Para cifrar el valor del texto sin cifrar 123, se calcula:

$$encrypt(123) = 123^{17} \pmod{3233} = 855$$

- Para descifrar el valor del texto cifrado, se calcula

$$decrypt(855) = 855^{2753} \pmod{3233} = 123$$

Ambos de estos cálculos pueden ser eficientemente usados por el algoritmo de multiplicación cuadrática para exponenciación modular.

Mediante “prueba y ensayo” es muy difícil calcular **d** ya que es un número de 512 bits o más. Así el sistema de criptoanálisis utilizado es buscar la clave privada **d**, a partir de las claves públicas **e** y **n**. Para esto basta con encontrar los números **p** y **q**; éstos son la descomposición en factores primos de **n**, ya que **n = p * q**. No se ha descubierto aún ninguna forma analítica de descomponer números grandes en factores primos.
Se consiguió descomponer una clave de 219 bits en el corto periodo de una semana (y Rivest perdió una apuesta). En agosto de 1999 la empresa RSA Inc. consiguió romper un RSA con clave de 512 bits, para ello necesitó 5.2 meses y 292 ordenadores entre PCs y estaciones de trabajo. Actualmente es aconsejable utilizar claves de 1024 bits. Está muy extendido como algoritmo asimétrico: es el más rápido y sencillo de los existentes.

Tiene todas las ventajas de los sistemas asimétricos, los servicios de autenticación y firma digital sólo se pueden implementar con estos sistemas. Para confidencialidad se puede utilizar también clave simétrica (DES, IDEA, RC4, etc.) y estos son mucho más rápidos que el RSA. En la actualidad se utilizan sistemas mixtos simétricos para confidencialidad y asimétricos para distribución de claves simétricas, autenticación y firma digital [TANENBAUN, 1997b].

2.3.3 Curvas Elípticas (CEE)

La encriptación de curvas elípticas es otro tipo de criptografía de clave pública, en el cual utiliza curvas elípticas definidas en un campo finito. La diferencia que existe entre este sistema y RSA es el problema matemático en el cual basan su seguridad. RSA razona en encontrando los factores primos. El problema en el cual están basados los sistemas que usan curvas elípticas es un logaritmo discreto elíptico.

A continuación se explica el funcionamiento de la CCE [ANGEL, 1999], [7]:

1. Se entiende como una curva elíptica a una ecuación de la forma siguiente:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

Donde las constantes a, b, c, d y e pertenecen a cierto conjunto llamado campo **F**, que para propósitos de la criptografía o es un campo primo (**Zp**) o un campo de característica 2, o sea donde los elementos son n-adas de ceros y unos (**F2n**).

2. A un punto que satisface la ecuación anterior se le llama punto racional. Si el campo es finito, entonces el conjunto de puntos (x, y) que satisfacen la ecuación es finito y es llamado conjunto de puntos racionales de la curva E sobre el campo **F**. Al conjunto de puntos racionales se puede representar como:

$$E : O, P_1, P_2, P_3, ..., P_n$$

E representa la ecuación y O es un punto que no tiene coordenadas y hace el papel de cero (llamado punto al infinito) ya que en este conjunto los puntos puede sumarse y tiene las mismas propiedades que la suma de los números enteros, es decir lo que se conoce como un grupo abeliano.

Ejemplo:

Una curva elíptica simple, si la ecuación es $y2 = x3 + 4x + 3$ y el campo **Z5**, es decir el conjunto $\{$**0, 1, 2, 3, 4**$\}$, entonces las parejas que satisfacen la ecuación son {**(2,2), (2,3)**}, por lo tanto la curva elíptica es E **:** {O ,**(2,2),(2,3)**}. En este caso E tiene 3 puntos racionales.

3. La suma de estos puntos tiene una explicación geométrica muy simple, si la gráfica representa a todos los puntos que satisfacen la ecuación de la curva elíptica, y se quiere sumar a P y Q, se traza una línea recta que pase por P y Q, la ecuación de la curva es de tercer grado y la línea de primer grado, entonces existen siempre tres soluciones, en este caso la tercera solución esta dibujada como el punto $-P-Q$, enseguida se procede a dibujar una línea recta paralela al eje Y que pase por $-P-Q$, esta línea vertical también intercepta tres veces a la recta, todas las líneas verticales interceptan al punto especial llamado infinito y que geométricamente esta en el horizonte del plano, el tercer punto es por definición $P+Q$, como se muestra en la figura 2.8:

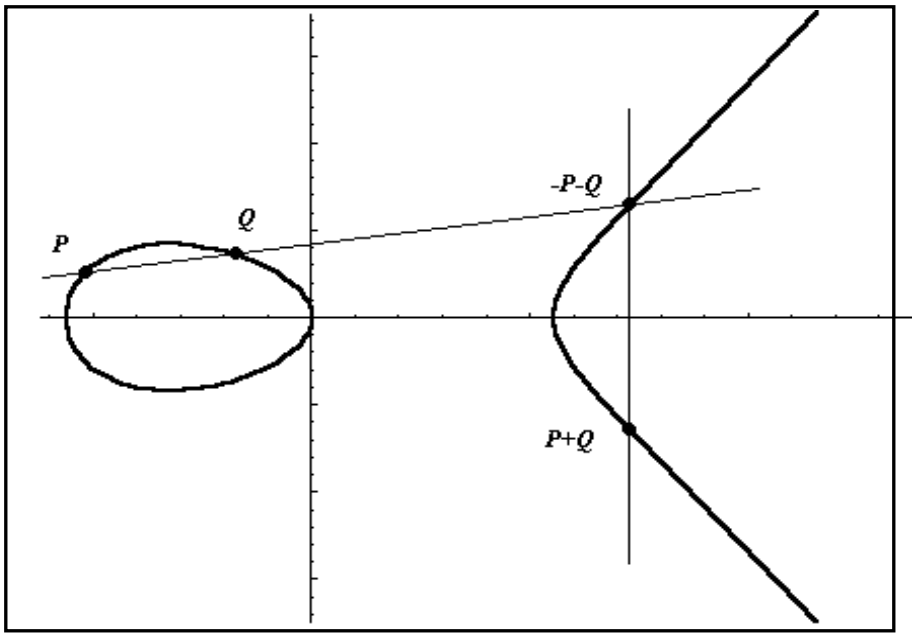

Figura 2.8 Graficación con los puntos P y Q

4. No es difícil obtener fórmulas para calcular las coordenadas del punto $P+Q$ a partir de las coordenadas del punto P y del punto Q. Por ejemplo si el campo de definición de la curva es un campo primo **Zp**, entonces las fórmulas de suma son las siguientes:

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1}, P = Q \end{cases}$$

5. La anterior forma de sumar puntos de una curva elíptica es un poco extraña no obstante, es esta extrañeza lo que permita que sea un poco más difícil romper los la encriptación de curvas elípticas. En el área de las matemáticas conocida como teoría de grupos se sabe que estos grupos son muy simples llamados grupo abelianos finitos lo que permite también que los CCE sean fácil de implementar, se llamara al número de puntos racionales de la curva como el orden de la curva.

Los CCE basan su seguridad en el Problema del Logaritmo Discreto Elíptico (PLDE), esto quiere decir que dados P, Q puntos de la curva hay que encontrar un número entero tal que $xP = Q$ ($xP = P + P + \ldots + P$, x veces). Obsérvese que a diferencia del PFE (Problema de Factorización Entera) el PLDE no maneja completamente números, lo que hace más complicado su solución.

La creación de un protocolo con criptografía de curvas elípticas requiere fundamentalmente una alta seguridad y una buena implementación, para el primer punto se requiere que la elección de la curva sea adecuada, el orden del grupo de puntos racionales tenga un factor primo de al menos 163 bits, además de que este orden no divida al orden de un número adecuado de extensiones del campo finito, para que no pueda ser sumergido en él, si el campo es **ZP**, se pide que la curva no sea anómala o sea que no tenga puntos P racionales. Todo esto con el fin de evitar los ataques conocidos.

Para el caso de la implementación hay que contar con buenos programas que realicen la aritmética del campo finito, además de buenos algoritmos que sumen

puntos racionales, tanto en el caso de **Zp** como **F2n**, en este último se toma una base polinomial que tenga el mínimo de términos por ejemplo un trinomio para generar los elementos del campo finito esto si la implementación es en software, y se toma una base normal si es en hardware. Además de contemplar que las operaciones de puntos racionales pueden hacerse en el espacio proyectivo, esto elimina el hacer divisiones, ahorrando tiempo.

Lo anterior se ve reflejado en las ventajas que ofrecen los CCE en comparación con RSA, la principal es la longitud de la clave secreta. Se puede mostrar que mientras en RSA se tiene que usar una clave de 1024 para ofrecer una considerable seguridad, los CCE solo usan 163 bits para ofrecer la misma seguridad, así también las claves RSA de 2048 son equivalentes en seguridad a 210 de CCE. Esto se debe a que para resolver el PLDE el único algoritmo conocido toma tiempo de ejecución totalmente exponencial, mientras que el algoritmo que resuelve PFE incluso también el PLD en Zp toma un tiempo subexponencial.

Otro aspecto importante es que los elementos de los puntos racionales pueden ser elementos de un campo finito de característica 2, es decir, pueden ser arreglos de ceros y unos de longitud finita (01001101110010010111), en este caso es posible construir una aritmética que optimice la rapidez y construir un circuito especial para esa aritmética, a esto se le conoce como Base Normal Optima.

Lo anterior permite con mucho que los CCE sean idóneos para ser implementados en donde el poder de cómputo y el espacio del circuito sea reducido, donde sea requerida una alta velocidad de procesamiento o grandes volúmenes de transacciones, donde el espacio de almacenamiento, la memoria o el ancho de banda sea limitado. Lo que permite su uso en *Smart Cards*, Teléfonos celulares, Fax, Organizadores de Palma, PCs, etcétera [ANGEL, 1999], [7].

2.4 FIRMA DIGITAL

Tradicionalmente, una validación de identificación de muchos documentos legales, financieros y otros tipos se determinan por la presencia o ausencia de una firma manuscrita autorizada. Dentro de los sistemas computarizados la firma digital con respecto a los documentos digitales es parecido a la de la firma de puño y letra (holográfica) en los documentos impresos: la firma es el sello irrefutable que permite atribuir a una persona algo escrito o su conformidad en un documento. El receptor, o un tercero, pueden verificar que el documento esté firmado, sin duda, por la persona cuya firma aparece en el documento y que éste no haya sufrido alteración alguna, el concepto de firma digital fue introducido por *Diffie* y *Hellman* en 1976.

La firma digital no implica que el mensaje esté encriptado, es decir, que este no pueda ser leído por otras personas; al igual que cuando se firma un documento holográficamente este puede ser visto por otras personas. Todo esto se resume en que la firma digital de un documento es el resultado de aplicar cierto algoritmo matemático, denominado función hash, a su contenido.

La funcionalidad de la firma digital se resume a cuatro aspectos:

1. **Autenticación**.- La firma digital es equivalente a la firma física de un documento.
2. **Integridad**.- El mensaje no podrá ser modificado.
3. **No repudio en origen**.- El emisor no puede negar haber enviado el mensaje.
4. **Encriptación de las comunicaciones**

Al igual que la criptografía, las firmas digitales se dividen en dos grandes grupos, firmas de clave secreta o y firmas de clave pública [TANENBAUN, 1997b], [7].

2.4.1 FIRMA DIGITAL CON CLAVE PRIVADA

Un enfoque de las firmas digitales será tener una autoridad central, la cual sepa todo y en quien todos confíen, en este caso X. Cada usuario escoge entonces una clave secreta y la lleva personalmente a las oficinas de X. Por tanto, sólo A y X conocen la clave secreta de A, K_A, etc.

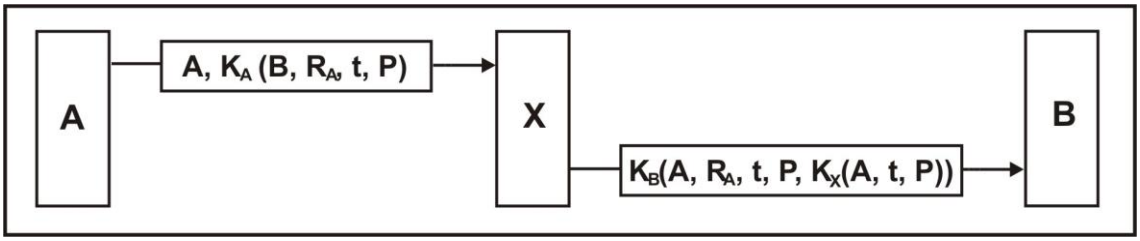

Figura 2.9 Firma Digital con Clave Secreta y Centro de Distribución de Claves

Cuando A quiere enviar un mensaje de texto normal firmado, P, a su banquero, B, genera K_A (B, R_A, t, P) y lo envía como se muestra en la figura 2.9. X ve que el mensaje es de A, lo descifra y envía un mensaje a B como se muestra. El mensaje a B contiene el texto normal del mensaje de A y también el mensaje firmado K_X (A, t, P), donde t es una marca de tiempo. Ahora B atiende la solicitud de A.

Ahora se ejemplificara que A niega el envío del mensaje, cuando el caso llega al juez y A niegue haber enviado a B el mensaje, B indica que el mensaje vino de A y no de un tercero C pues X no hubiera aceptado un mensaje de A, a menos que estuviese cifrado con K_A, por lo que no hay posibilidad de que C enviará a X un mensaje falso en nombre de A. B además presenta la prueba K_X(A, t, P). Entonces el juez pide a X (en quien todo el mundo confía) que descifre la prueba. Cuando X testifica que B dice la verdad el caso queda resuelto.

Un problema potencial del protocolo de firma anterior es que C repita cualquiera de los dos mensajes. Para minimizar este problema, se usan en todos los intercambios marcas de tiempo. Es más, B puede revisar todos los mensajes recientes para ver si se usó R_A en cualquiera de ellos. De ser así, el mensaje se

descarta como repetición. Nótese que B rechazará los mensajes muy viejos con base en la marca de tiempo. Para protegerse contra ataques de repetición instantánea, B simplemente examina el R_A de cada mensaje de entrada para ver si un mensaje igual se recibió de A durante el tiempo de validez de la marca temporal. Si no, B puede suponer con seguridad que ésta es una solicitud nueva [TANENBAUN, 1997b].

2.4.2 FIRMA DIGITAL CON CLAVE PÚBLICA

Una firma digital es la combinación de un resumen del texto encriptado con la clave privada del emisor, que se adjunta al documento que se firma, y constituye un método seguro de garantizar autoría y emisión del mismo [ÁNGEL, 1999], [OPPLIEGER, 1988], [CABALLERO, 1996].

El proceso de firma digital consta de dos partes bien diferenciadas [ÁNGEL, 1999], [CABALLERO, 1996], [ROBLINNG, 1982], [7]:

1. **Proceso de Firma**: Proceso donde el emisor encripta el documento con su llave privada y realiza un resumen del documento cuyo criptograma es la firma del emisor. Así el emisor no puede negar la procedencia ya que se ha encriptado con su clave privada, enviando al destinatario tanto el documento en claro como el encriptado como se observa en la parte superior de la figura 2.10.

2. **Proceso de Verificación de la Firma**: El receptor desencripta el documento cifrado con la clave pública de A y comprueba que coincide con el documento original, lo que prueba de forma total que el emisor del mismo ha sido efectivamente A. Por otro lado el receptor no puede modificar el contenido debido a que el resumen sería diferente y se percibiría que no coincide con la desencriptación de la firma como se observa en la parte inferior de la figura 2.10.

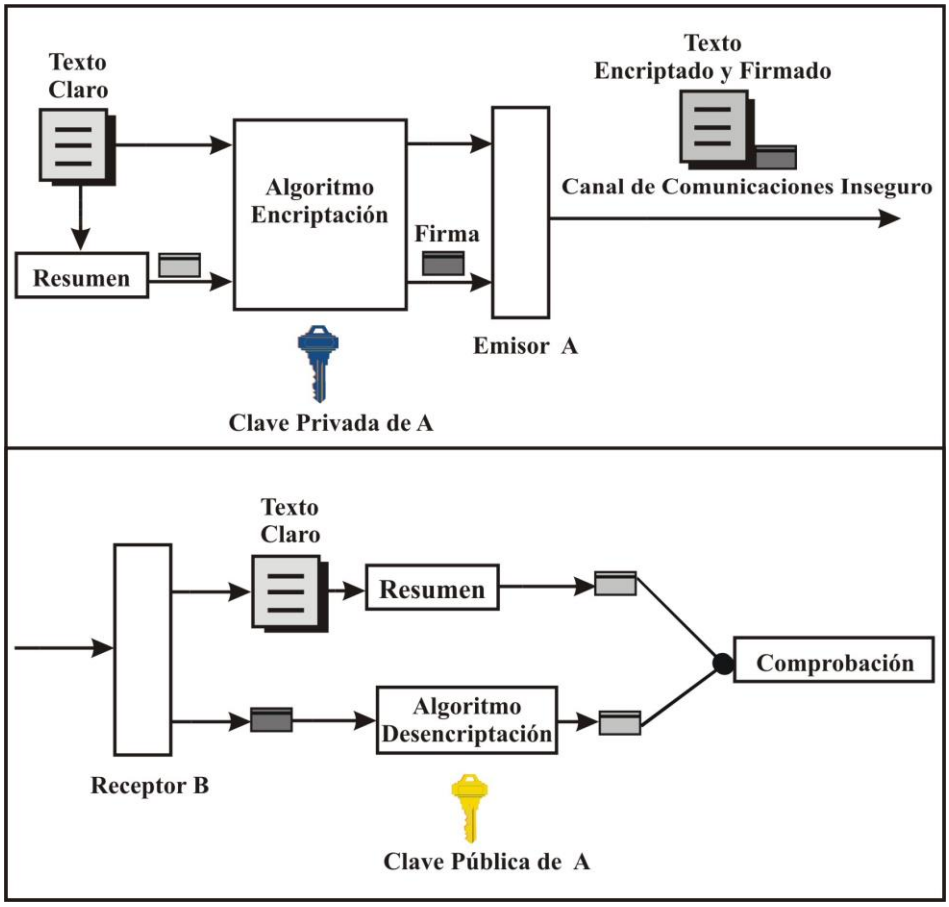

Figura 2.10 Esquema Básica de una Firma Digital

Como lo muestra la figura 2.10 el método de la firma digital no sólo ofrece autenticidad al mensaje enviado por A, además asegura el no repudio, ya que sólo el dueño de una llave privada puede encriptar un documento de manera que se pueda desencriptar con su llave pública, lo que asegura que ha sido A y no otro el que ha enviado dicho documento. Así mismo proporciona Integridad de datos, ya que si el documento fuera accedido y modificado en el camino el resumen del documento cambiaría también.

La firma digital suele usarse en comunicaciones en las que no existe una confianza inicial total entre los comunicantes. Se usan para autentificar mensajes, para validar compras por Internet, para realizar transferencias de fondos bancarios y para otras transacciones de negocios [ROBLINNG, 1982], [MAYEL; MATYAS, 1982], [7].

2.5 FUNCIONES *HASH*

Una herramienta esencial en la criptografía, son las funciones *hash*. Son usadas principalmente para resolver el problema de la integridad de los mensajes, así como la autenticidad de mensajes y de su origen [ÁNGEL, 1999].

Por ejemplo, si se imagina el envío de un documento extenso que se quiere firmar digitalmente, se podrá notar que cifrar el documento enteramente es una pérdida de tiempo, ya que los medios de encriptación de llave pública son lentos, debido a que requieren generalmente operaciones aritméticas demasiado complejas, teniendo una capacidad de proceso limitada dando como resultado procesos demasiado lentos. Y para resolver éste problema aparecen las funciones *hash*, que son unas funciones matemáticas que realizan un resumen del documento a firmar. Su forma de operar es comprimir el documento en un único bloque de longitud fija, bloque cuyo contenido es ilegible y no tiene ningún sentido real. A consecuencia de lo anterior, por definición las funciones *hash* son irreversibles, es decir, que a partir de un bloque comprimido no se puede obtener el bloque sin comprimir, y si no es así no es una función *hash*. Estas funciones son además de dominio público [CABALLERO, 1996], [MAYEL; MATYAS, 1982], [7].

A un mensaje resumido mediante una función *hash* y encriptado con una llave privada es lo que en la vida real se denomina **firma digital** descrita anteriormente. Como se observar en la figura 2.11, el emisor A, utiliza una función *hash* conocida al documento, posteriormente encripta dicho resumen con su clave privada, envía al receptor el documento original plano y el resumen *hash* encriptado. Después el receptor B aplica la función *hash* al resumen sin encriptar y desencripta el resumen encriptado con la llave pública de A. Y si ambos coinciden está seguro de que ha sido A quien ha enviado el documento. Si no coinciden, está seguro de que no ha sido A o de que el envío ha sido interceptado durante el medio de envío y modificado. El caso de que ambos resúmenes no coincidan contempla también la posibilidad de que el mensaje haya sido alterado en su viaje de A hacia B, lo que

conlleva igualmente el rechazo del documento por no válido. [ÁNGEL, 1999], [ROBLINNG, 1982], [7].

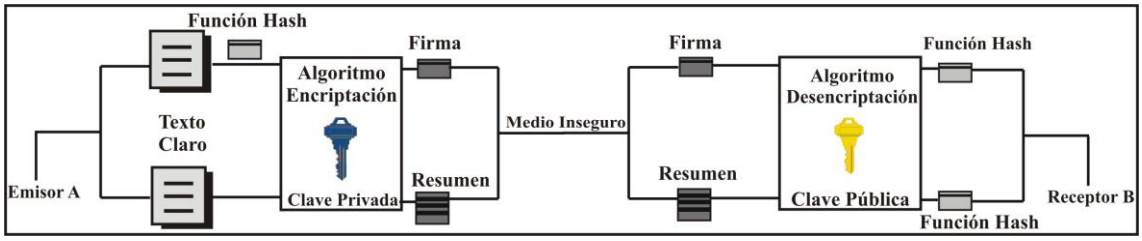

Figura 2.11 Esquema de Firma Digital Mediante una Función *Hash*

Las funciones *hash* y la firma digital son elementos indispensables para establecer canales seguros de comunicación, basados en los Certificados Digitales.

Para que una función pueda considerarse como función *hash* debe cumplir con lo siguiente [ROBLINNG, 1982]:

- Transformar un texto de longitud variable en un bloque de longitud fija, que generalmente es pequeña (algunas son de 16 bits).

- Ser cómoda de usar e implementar.

- Irreversible, es decir, no se puede obtener el texto original del resumen *hash*.

- Debe ser imposible encontrar dos mensajes diferentes cuya firma digital mediante la función *hash* sea la misma (no-colisión).

- Si se desea además mantener un intercambio de información con Confidencialidad, basta con cifrar el documento a enviar con la clave pública del receptor.

Las funciones *hash* más conocidas y usadas son [ÁNGEL, 1999], [MAYEL; MATYAS, 1982], [OPPLIEGER, 1988], [6]:

MD2.- Abreviatura de *Message Digest 2*, diseñado para computadoras con procesador de 8 bits. Todavía se usa, pero no es recomendable, debido a su lentitud de proceso.

MD4.- Abreviatura de *Message Digest 4*, desarrollado por *Ron Rivest*, uno de los fundadores de RSA Data Security Inc. y padre del sistema asimétrico RSA. Aunque se considera un sistema inseguro, es importante porque ha servido de base para la creación de otras funciones *hash*. Un sistema de ataque desarrollado por *Hans Dobbertin* posibilita el crear mensajes aleatorios con los mismos valores de *hash* (colisiones), por lo que ya no se usa. De hecho, existe un algoritmo que encuentra una colisión en segundos.

MD5.- Abreviatura de *Message Digest 5*, también obra de *Ron Rivest*, que se creó para dar seguridad a MD4, y que ha sido ampliamente usado en diversos campos, como autenticador de mensajes en el protocolo SSL y como firmador de mensajes en el programa de correo PGP. Sin embargo, fue reventado en 1996 por el mismo investigador que lo hizo con MD4, el señor *Dobbertin*, que consiguió crear colisiones en el sistema MD5, aunque por medio de ataques parciales. Pero lo peor es que también consiguió realizar ataques que comprometían la no-colisión, por lo que se podían obtener mensajes con igual *hash* que otro determinado. A pesar de todo esto, MD5 se sigue usando bastante en la actualidad.

SHA-1.- *Secure Hash Algorithm*, desarrollado como parte integrante del *Secure Hash Standar* (SHS) y el *Digital Signature Standar* (DSS) por la Agencia de Seguridad Nacional Norteamericana, NSA. Sus creadores afirman que la base de este sistema es similar a la de MD4 de *Rivest*, y ha sido mejorado debido a ataques nunca desvelados. La versión actual se considera segura (por lo menos hasta que se demuestre lo contrario) y es muy utilizada algoritmo de firma, como

en el programa PGP en sus nuevas claves DH/DSS (*Diffie-Hellman/Digital Signature Standar*). Destacar también que en la actualidad se están estudiando versiones de SHA con longitudes de clave de 256, 384 y 512 bits.

RIPEMD-160.- desarrollada por un grupo de investigadores europeos, entre los que se encuentra *Hans Dobbertin* (el reventador de MD4-MD5) y otros investigadores incluidos en el proyecto RIPE (*RACE Integrity Primitives Evaluation*). Su primera versión adolecía de las mismas debilidades que MD4, produciendo colisiones, pero las versiones mejoradas actuales son consideradas seguras. Maneja claves muy robustas, normalmente de 160 bits, aunque existen versiones de 128 y se están planteando nuevas de 256 y 320 bits. Es muy rápido, no está patentado y su código fuente es abierto, de libre acceso.

CAPITULO III

KERBEROS

3.1 INTRODUCCIÓN

Cuenta la mitología griega acerca de un perro de tres cabezas, llamado **Kerberos** (en español Cancerbero), el cual vigilaba las puertas del infierno. Su función de **Kerberos** era dejar entrar con facilidad a las almas que permanecerían ahí y de aniquilar a quien no tuviesen autorización de penetrar en el infierno.

Como sucede en algunas películas de personas que burlan la seguridad de los sistemas computacionales, hoy en día ocurre con mucha frecuencia dejando de lado la ciencia ficción, en las revistas de interés general o periódicos se publica constantemente información acerca de fraudes informáticos multimillonarios. Los *hackers* han encontrado múltiples formas de traficar con la información ajena, desde la transferencia directa de fondos, hasta la venta de información confidencial de una compañía a otra. Incluso, existen "*Hackers* profesionales", los cuales son contratados por compañías para que intenten romper sus sistemas de seguridad y con ello saber si efectivamente están protegidas o no contra este tipo de ataque.

El manejo de la seguridad no es únicamente para agencias militares o gubernamentales, sino también para situaciones tales como usuarios no autorizados que intentan revisar documentos, copiar o robar archivos y *passwords*, modificar datos o leer y utilizar documentos de correo electrónico sin autorización.

Lo que era un hecho al utilizar el ambiente cliente/servidor y una tendencia a la utilización del ciberespacio traía consigo nuevos requerimientos de seguridad que no existían en los ambientes no distribuidos. En los sistemas no distribuidos se

podía confiar en la autorización de acceso a los sistemas operativos. En el caso de los sistemas distribuidos el reto era mucho mayor ya que la información fluye a través de una red y esta puede ser observada e interceptada. Se requería de nuevas formas de seguridad para proteger los servidores de accesos no autorizados. El reto era lograr un alto nivel de protección y seguridad con el mayor grado de transparencia de estos mecanismos de control para el usuario autorizado.

A inicios de los años ochenta, algunos investigadores del MIT, trabajaban en un proyecto de redes llamado *Athena*, que consistía en integrar múltiples computadoras heterogéneas en una red de alto rendimiento y alta velocidad. La necesidad de integra una red con una gran cantidad de computadoras de las cuales la mayoría eran PC's, inmediatamente despertó preocupación con relación a cómo podrían evitar que los estudiantes o usuarios externos ejecutaran programas que espiaran o dañaran la información de la red. Llegando a la conclusión de que en un ambiente distribuido es casi imposible asegurar que todas las estaciones de trabajo en una red sean seguras. Basados en lo anterior diseñaron un esquema de seguridad al cual llamaron **Kerberos**.

Al igual que el monstruo mitológico el mecanismo de seguridad de **Kerberos** consiste en verificar la identidad de los usuarios, cliente o servidores (llamados unidades de confianza) y permitir anular acceso a los servicios de red. **Kerberos** requiere que las unidades de confianza proporcionen su contraseña que compruebe su identidad. El proceso que utiliza es semejante a lo que sucede en las películas de espías para identificarse entre sí. Un espía que personifica a un recepcionista de un hotel se le acerca a un agente que se va a registrar en dicho hotel y le dice: "Parece que lloverá en el Amazonas". El agente le contesta:" Desafortunadamente no traigo zapatos cafés". A través de este intercambio de frases en clave, ambos espías verifican la identidad del otro y obtiene el suficiente nivel de confianza para continuar con sus asuntos.

En el caso de los sistemas computacionales de una red se deben establecer altos niveles de confianza entre los diferentes elementos antes de que haya intercambio de información ya que además de tener el riesgo de usuarios no autorizados, existe el peligro de la usurpación de servidores o de recursos. Por ejemplo, si se habla de un usuario que desea imprimir información confidencial en una impresora de red, donde los sistemas de seguridad no hace una verificación del que el dispositivo al cual va la información sea efectivamente una impresora, esta puede ser usurpada por una máquina de fax que este imprimiendo la información confidencial en las oficinas de un competidor [RED, 1995].

3.2 DEFINICIÓN

Kerberos es un servicio de autentificación de red. Se diseño para lograr un autentificación real en las aplicaciones cliente-servidor utilizando la criptografía y así restringir los accesos sólo a usuarios autorizados y autentificar los servidores a los cuales se les solicita algún servicio, asumiendo un entorno distribuido abierto, en el cual los usuarios a través de sus estaciones de trabajo acceden a servicios en servidores distribuidos a través de una red.

En el entorno de trabajo que considera **Kerberos**, una estación de trabajo no puede ser confiable para identificar a sus usuarios correctamente para servicios de red. En particular, puede suceder lo siguiente:

- Un usuario puede ganar el acceso a una estación de trabajo en particular y aparentar ser otro usuario operando desde otra estación de trabajo.
- Un usuario puede alterar la dirección de red de una estación de trabajo para que los requerimientos que envía simulen llegar desde otra estación de trabajo.
- Un usuario puede "escuchar disimuladamente" los intercambios y atacar para ganar la entrada a un servidor o para interrumpir operaciones.

En muchos de esos casos, un usuario podría ganar acceso a servicios y datos que no está autorizado a acceder. En lugar de construir protocolos de autenticación en cada servidor, **Kerberos** provee un servidor de autentificación centralizado, cuya función es autenticar usuarios frente a servidores y servidores frente a usuarios. **Kerberos** usa cifrado simétrico (también llamado cifrado convencional), método en el cual el cifrado y descifrado se realizan usando una única clave [12] [13].

Existe un dialogo que provee un relato ficticio del diseño de un sistema de autentificación para redes abiertas llamado "*Charon*". En el transcurso del dialogo, los personajes *Athena* y *Eurípide*s descubren los problemas de seguridad inherentes en un ambiente de red abierto.

Cuando terminan de diseñar el sistema, *Athena* le cambia el nombre a "**Kerberos**", coincidentemente el mismo nombre del sistema de autentificación que fue diseñado en el proyecto *Athena* del MIT.
Existe un diálogo llamado "Diseñando un Sistema de la Autenticación: Un Dialogo en Cuatro Escenas", fue escrito en febrero de1988 por *Bill Bryant* para ayudar a sus lectores a entender las razones fundamentales por las cuales el protocolo **Kerberos** V4 estaba implementado de esa manera, dicho diálogo ofrece un excelente enfoque del sistema de autentificación **Kerberos**, el cual se podrá observar en el anexo 1 [14], [15].

3.3 DESARROLLO

El sistema de autentificación y de distribución de claves **Kerberos** fue desarrollado por en el MIT (Instituto de Tecnología de Massachussets) en colaboración de IBM y con *Digital Equipment Corparation* y diseñado por *Miller y Neuman* para proteger los servicios de red que nacieron con el proyecto *Athena*. Está basado en el protocolo de distribución de claves presentado por *Needham y Schroeder* en 1978 [OPPLIEGER, 1998], [12], [18].

El objetivo de **Kerberos** fue ampliar el conocimiento de autentificación, autorización y contabilidad del entorno de computación y de red de MIT. De acuerdo al plan técnico del proyecto Athena, este entono estaba formado básicamente por:

- Estaciones de trabajo públicas y privadas.

    1. Las estaciones públicas se encuentran en áreas sin seguridad ó sólo seguridad mínima.

    2. Las estaciones privadas están bajo control físico y administrativo de individuos que por lo regular no tienen idea de las responsabilidades de un administrador de red.

- Una red en el campus de la universidad, por redes de área local (LAN – *Local Area Networks*) de varios tipos conectados a una red troncal. Las redes LAN se encontraban implementadas en instalaciones geográficamente dispersas y eran vulnerables a diversos ataques. En cambio la red troncal esta resguardada en armarios por tanto en condiciones de seguridad física moderada.

- Servidores operados centralmente. La mayoría de dicho servidores se encontraban en habitaciones cerradas, y por consiguiente en condiciones de seguridad física moderada, con *software* que no contiene código malicioso. Sólo algunos servidores funcionaban en condiciones de considerable seguridad física, y se podían utilizar como servidores de seguridad.

Como se puede observar en dicho entorno, no es adecuado para almacenar, procesar, o transmitir datos significativos, como registros financieros o datos clasificados. Los riesgos surgen como consecuencia del uso incontrolado de

recursos por parte de entidades no autorizadas, violaciones de la integridad de los recursos del sistema y los atentados masivos a la intimidad, como por ejemplo, las visualizaciones los archivos personales de cada usuario. Las amenazas a la seguridad aparecen de la posibilidad de que un usuario de una estación de trabajo falsifique la identidad de otro para obtener accesos no autorizados a los sistemas [COULOURIS, et al., 1994], [OPPLIEGER, 1998].

Las primeras tres versiones de **Kerberos** solo se usaron en el MIT. Por lo que la primera versión disponible para el público fue la versión 4 (V4). Esta versión se utilizo ampliamente en el MIT. Y aun que el MIT no preveía nuevas versiones, en algunos lugares que lo utilizaban, requerían servicios que **Kerberos** no proporcionaba. Como resultado se produjo la versión 5 de **Kerberos** (V5).

Hasta que se diseñó **Kerberos**, la autenticación en redes de computadoras se realizaba principalmente de dos formas:

- **La autenticación por declaración** (*Authentication by assertion*), en la que el usuario es libre de indicar el servicio al que desea acceder (por ejemplo, mediante el uso de un cliente determinado).

- Utilizaban **contraseñas** para cada servicio de red.

Evidentemente el primer modelo proporciona un nivel de seguridad muy bajo, ya que se le otorga demasiado poder al cliente sobre el servidor, y el segundo modelo tampoco es muy conveniente: debido a que por un lado se obliga al usuario a ir tecleando constantemente su clave, de forma que se pierde demasiado tiempo y además la contraseña está viajando continuamente por la red [17].

3.4 PROTOCOLO DE *NEEDHAM – SCHOREDER*

Kerberos se basa en los protocolos de autentificación y distribución de claves propuestos originalmente por *Needham* y *Schroeder*. Dicho protocolo supone la existencia de una tercera identidad de confianza que funciona como servidor de autentificación (*AS*). El *AS* comparte un clave secreta con cada principal, dos principales arbitrarios *A* y *B* pueden utilizar el *AS* para autentificarse mutuamente y para proporcionarse una clave de sesión K_{AB}. El **primer protocolo** de *Needham* y *Schroeder* se establecen en cinco pasos como se muestra a continuación en la tabla 3.1 [OPPLIEGER, 1998], [23], [24].

Paso 1	$A \rightarrow AS : A, B, N_A$
Paso 2	$AS \rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\} K_B\} K_A$
Paso 3	$A \rightarrow B : \{K_{AB}, A\} K_B$
Paso 4	$B \rightarrow A : \{N_B\} K_{AB}$
Paso 5	$A \rightarrow B : \{N_B - 1\} K_{AB}$

Tabla 3.1 Primer Protocolo de *Needham* y *Schroeder*

Notación:

AS	Servidor de autentificación.
A	Nombre del principal que inicia el proceso de comunicación.
B	Nombre del principal para el compañero de comunicación de A.
N_A	Valor temporal generado por A.
K_A	Clave de sesión de A.
K_B	Clave de sesión de B.
K_{AB}	Clave de sesión para la comunicación entre A y B.
N_B	Valor temporal generado por B.

En el **paso 1,** el protocolo inicia cuando *A* comunica al *AS* (sin encriptar) su identidad y la de *B***,** incluye también un valor temporal, N_A. Después de recibir el

mensaje, *AS* busca la clave secreta de *A* y *B*, es decir K_A y K_B, posteriormente selecciona aleatoriamente una clave de sesión K_{AB} que deberán compartir *A* y *B*. En **paso 2**, *AS* regresa a, $A\ \{N, B, K_{AB}, \{K_{AB}, A\} K_B\} K_A$, conteniendo la nueva clave generada K_{AB}, como también un *ticket* encriptado con la clave de sesión de *B*, el valor N_A demuestra que el mensaje fue enviado en respuesta al primer paso y *A* confía que fue enviado por *AS* porque solo *AS* sabe K_A. Como consecuencia de que el mensaje esta encriptado con K_A solo *A* será capaz de desencriptar y extraerlo. En el **paso 3**, A envía el *ticket* ($\{K_{AB}, A\} K_B$) a *B* y debido a que esta encriptado con K_B solo *B* podrá desencriptarlo y extraer la clave de sesión K_{AB}, así como también podrá saber la identidad de *A* autentificado por el *AS*. En el **paso 4**, para protegerse de un ataque, *B* elige un valor temporal N_B, encriptándolo con K_{AB}, enviándoselo a, *A*. Y por ultimo en el **paso 5** A regresa con el valor temporal decrementado y cifrado. De esta manera, si *B* recibe la respuesta satisfactoriamente, la confianza es mutua y es suficiente para permitir que comience la comunicación encriptada con K_{AB} [COULOURIS, et al., 1994], [LAM; BHET, 1992], [OPPLIEGER, 1998], [23]. En el primer protocolo de *Needham* y *Schroeder* consta de cinco pasos. Este número puede reducirse a tres si *A* puede almacenar para sus interlocutores de comunicación elementos de la forma *B*, K_{AB}, $\{K_{AB}, A\} K_B$, de esta manera se elimina los pasos 1 y 2. Pero si se almacenan dichos autenticadores, se tiene que realizar cambios al protocolo. Debido a esto los reconocimientos de los mensajes son bilaterales. El protocolo resultante es el **segundo protocolo** de *Needham* y *Schroeder* como se presenta a continuación en la tabla 3.2.

Paso 1	$A \rightarrow AS : A, B, N_A$
Paso 2	$AS \rightarrow A : \{N_A, B, K_{AB}, \{K_{AB}, A\} K_B\} K_A$
Paso 3	$A \rightarrow B : \{K_{AB}, A\} K_B, \{N_A\} K_{AB}$
Paso 4	$B \rightarrow A : \{N_B - 1\} K_{AB}$
Paso 5	$A \rightarrow B : \{N_B - 1\} K_{AB}$

Tabla 3.2 Segundo Protocolo de *Needham* y *Schroeder*

Una de las principales debilidades de las dos versiones de este protocolo, la causa la posibilidad de que un intruso *C*, sea capaz de interceptar el flujo de mensajes y posteriormente realizar un ataque sobre K_{AB}. Si *C* es capaz de determinar una clave de sesión K_{AB} que se haya usado en el pasado, *C* podrá suplantar a, *A*. De esta manera C puede replicar el billete correspondiente junto con el valor temporal cifrado con K_{AB} en el paso 3. *B* enviaría a *A* el valor temporal en el paso 4, pero como *C* conoce K_{AB}, podría responder correctamente en el paso 5: La debilidad en la que reside todo el problema es que el mensaje que se envía en el paso 3 no contiene información para que verifique su vigencia. En realidad, solo *A* y *AS* pueden saber si K_{AB} es reciente. Este problema fue descubierto por *Denning* y *Sacco*. Debido a lo anterior, propusieron reemplazar los valores temporales por marcas temporales y utilizar el protocolo que se muestra en la tabla 3.3 [COULOURIS, et al., 1994], [LAM; BHET, 1992], [OPPLIEGER, 1998], [23].

Paso 1	$A \rightarrow AS : A, B$
Paso 2	$AS \rightarrow A : \{B, K_{AB}, T, \{K_{AB}, A, T\} K_B\} K_A$
Paso 3	$A \rightarrow B : \{K_{AB}, A, T\} K_B$
Paso 4	$B \rightarrow A : \{N_B\} K_{AB}$
Paso 5	$A \rightarrow B : \{N_B - 1\} K_{AB}$

Tabla 3.3 Protocolo de *Denning* y *Sacco*

Los Protocolos tienen más similitudes que diferencias. El protocolo *Denning* y *Sacco*, *A* manda *A* y *B* al *AS* en el **paso 1**, y *AS* devuelve $\{B, K_{AB}, T, \{K_{AB}, A, T\} K_B\} K_A$ a, *A* en el **paso 2**. En este caso T representa una marca temporal. Nuevamente el mensaje se encuentra cifrado con K_A. *A* puede descifrar este mensaje y extraer en consecuencia *B*, K_{AB}, *T*, $\{K_{AB}, A, T\} K_B$. *A* valida *T* y si la marca temporal es válida, envía a *B* el *ticket* ($\{K_{AB}, A, T\} K_B$) en el **paso 3**. B puede descifrar el *ticket* y obtener en consecuencia K_{AB}, *A* y *T*. *B* puede también validar la marca temporal *T*. Si *B* considera que *T* es reciente, selecciona otro valor temporal N_B, lo encripta K_{AB} y devuelve a, $A \{N_B\} K_{AB}$ en el **paso 4**. *A* debe

finalmente desencriptar N_B, decrementándolo, volverlo a encriptar con la misma clave de sesión y devolver $B\ \{N-1\}\ K_{AB}$ en el **paso 5** [OPPLIEGER, 1998].

3.5 ARQUITECTURA

En la terminología de **Kerberos**, los dominios de administración se denominan **reinos**. Se supone que cada compañía u organización que quiera utilizar **Kerberos** establecerá un reino determinado por un **nombre de reino**. En teoría cada reino puede admitir hasta 100 000 usuarios. Como el reino ATENÍA.MIT.EDU que en la actualidad tiene alrededor de 25 000 usuarios, de los cuales unos 7 000 se conectan cada día [OPPLIEGER, 1998].

Kerberos se basa en el modelo cliente/servidor, Los usuarios, los clientes y los servicios de red ejecutándose en sistemas concretos se consideran generalmente **principales**. Cada principal se identifica mediante un único identificador. Dicho identificador del principal está formado por tres componentes [OPPLIEGER, 1998], [18], [19]:

1. El nombre de **principal,** es único para cada cliente, usuario y servicio asignado por el Administrador de **Kerberos**.

2. El nombre de **instancia**, algunos otros autores lo denominan de realización, usado para la autentificación, es una etiqueta añadida para clientes y servicios que existe bajo diversas formas. Para usuarios, puede proporcionar diferentes instancias para máquinas diferentes. Para servicios, una instancia suele especificar el nombre del *host* en la máquina que proporciona el servicio. También puede especificar los privilegios de cada uno.

3. El nombre de **reino,** nombre de una entidad administrativa que mantiene datos de autenticación. El nombre principal y el de instancia son calificados

por el del reino al que pertenecen, y son únicos sólo dentro de ese reino. El reino es habitualmente el nombre del dominio.

El objetivo de **Kerberos** es permitir que un cliente ejecutándose en nombre de un usuario pruebe su identidad a un servicio o a un servidor de aplicaciones sin necesidad de enviar datos por la red, ya que podría facilitar a un agresor suplantar posteriormente a un usuario. Para conseguir esto, el modelo **Kerberos** necesita la existencia de una tercera entidad de confianza que actué como **centro de distribución de llaves** (*KDC – key distribution center*) en el reino de **Kerberos**. *KDC* consta de dos componentes fundamentales:

1. Un servidor de autentificación (*AS - Authentication Service*)

2. Servidor de Emisión de *Tickets* (*TGS - Ticket Granting Service*)

El primero tiene como función autenticar inicialmente a los clientes y proporcionarles un ticket para comunicarse con el segundo, el servidor de *tickets*, que proporcionará a los clientes las credenciales necesarias para comunicarse con un servidor final que es quien realmente ofrece un servicio. Además, el servidor posee una base de datos de sus clientes (usuarios o programas) con sus respectivas claves privadas, conocidas únicamente por *KDC* y por el cliente que al que pertenece [COULOURIS, et al., 1994], [Opplieger, 1998], [21], [22].

La arquitectura de **Kerberos** está basada en tres objetos de seguridad [COULOURIS, et al., 1994], [19], [23], [24]:

1. **La clave de sesión** es una clave secreta generada por **Kerberos** y expedida a un cliente para comunicarse con un servidor durante una sesión; no es obligatorio utilizarla en toda la comunicación con el servidor, sólo si el servidor lo requiere (porque los datos son confidenciales) o si el servidor es un servidor de autenticación. Como también no es obligatoria la

encriptación en toda la comunicación con el servidor; la clave de sesión es usada para encriptar la comunicación con los servidores que lo requieran y para encriptar todos los autenticadores. Se suele denominar a esta clave K_{CS}, para la comunicación entre un cliente *C* y un servidor *S*. Las claves de sesión se utilizan para minimizar el uso de las claves secretas de los diferentes agentes: éstas últimas son válidas durante mucho tiempo, por lo que es conveniente utilizarlas lo menos posible para minimizar la posibilidad de un ataque.

2. **El ticket** es un testigo expedido a un cliente por servidor de emisión de *tickets* de **Kerberos** para solicitar los servicios de un servidor; garantiza que el cliente ha sido autenticado recientemente. El ticket de un cliente *C* para acceder a un servicio *S* se le denomina $\{ticket(C, S)\}_{K_s} = \{C, S, t_1, t_2, K_{CS}\}_{K_s}$. Este ticket incluye el nombre del cliente *C*, para evitar su posible uso por impostores, un periodo de validez comenzando con la fecha de emisión del *ticket* t_1 y la fecha de expiración t_2 y una clave de sesión K_{CS} asociada para uso de cliente y servidor. **Kerberos** siempre proporciona el ticket ya cifrado con la clave secreta del servidor al que se le entrega.
3. **El autentificador o autenticador** es un testigo construido por el cliente y enviado a un servidor para probar su identidad y la actualización de cualquier comunicación con el servidor; sólo puede ser utilizado una vez. Un autenticador de un cliente *C* ante un servidor *S* se denota por $\{auth(C)\}_{K_{cs}} = \{C, t\}_{K_{cs}}$. Este autenticador contiene, el nombre del cliente y un *timestamp*, y es encriptada con la apropiada clave de la sesión.

La figura 3.1 ilustra el modelo básico de **Kerberos** y los correspondientes pasos, algunos autores definen 4 etapas como se explicara a continuación. La situación comienza en el lado izquierdo, cuando un cliente se está ejecutando en el nombre de un usuario. Para poder utilizar los servicios del servidor que se muestra en la parte inferior derecha de la figura y el cliente debe autentificarse ante dicho servidor.

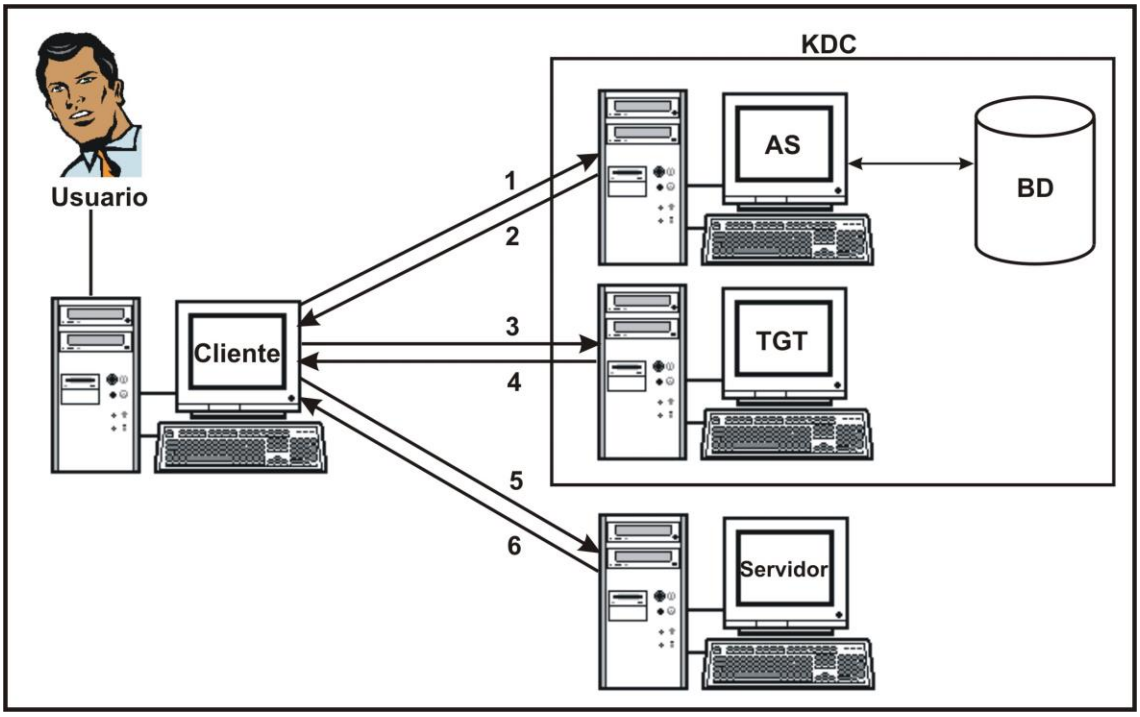

Figura 3.1 Modelo Básico de Kerberos

Como se puede observar en los pasos de la figura 3.1, cuando un usuario va a una estación de trabajo e intenta ingresar, debe proporcionar como siempre su nombre de usuario. El cliente en representación del usuario proporciona al *AS* el nombre del usuario en el **paso 1**, y el *AS* busca la identidad de dicho usuario en la base de datos del *KDC*. La clave de sesión y un ticket enviadas por el *AS* en el **paso 2**. El cliente le debe pedir al usuario que introduzca su contraseña o *password* que es necesario para descifrar el *TGT*. Solo si clave de sesión obtenida a partir de la contraseña del usuario descifra adecuadamente el *TGT*, el usuario podrá entrar, de lo contrario se le permitirá su ingreso.

En la parte del cliente, el *TGT* se guarda para su uso futuro, para obtener los *tickets* que se emplearan para autentificar servicios de red. El principal propósito de *TGT* es proporcionar a los usuarios una utilidad de registro único. El usuario solo introduce su contraseña o *password* solo una vez y no es requerido cada vez que se solicita un servicio.

Para obtener un *ticket*, el cliente envía la respectiva solicitud al *TGT* en el **paso 3**. El mensaje contiene el nombre del servicio solicitado, el TGT y un autentificador. Si el *TGT* considera que el ticket y el autentificador son validos, en el **paso 4** se devuelve un *ticket* y una clave de sesión para el servicio de red solicitada. El cliente construye otro autentificador y lo envía junto con ticket de servicio en el **paso 5** al servidor que proporcionara el servicio de red deseado. Y por ultimo en el paso re requiere una autentificación mutua, el servidor le devuelve al cliente un autentificador en el **paso 6** [COULOURIS, et al., 1994], [HALSALL, 1998], [OPPLIEGER, 1998], [19], [23], [24].

De manera general el funcionamiento de **Kerberos** puede ser dividido en cuatro etapas [25]:

1. Conexión al sistema: un cliente se autentica frente a **Kerberos** recibiendo un *ticket* y una clave de sesión correspondiente al intercambio entre el cliente y **Kerberos**, encriptados con una clave secreta derivada del *password* del usuario.
2. Solicitud de acceso al servidor: para cada nuevo servicio a ser utilizado por el cliente debe ser solicitado un *ticket* y una clave de sesión para ese nuevo intercambio entre el cliente y el servidor. Esa información es enviada al cliente encriptada con una clave de sesión entre el cliente y **Kerberos**.
3. Acceso al servidor: el cliente inicialmente envía el *ticket* recibido para el intercambio con el servidor. Debido a que ese *ticket* está encriptado con una clave secreta del servidor, el mismo deberá desencriptarla y así tendrá acceso a la clave de sesión entre el cliente y el servidor, que será válida por un período limitado (tiempo de vida), que generalmente es de 8 horas. Después de este procedimiento, la comunicación entre ellos podrá ser llevada a cabo de una manera confidencial.
4. Mantenimiento de la Base de Datos de **Kerberos**: comprende el almacenamiento de las claves secretas del cliente y del servidor.

3.6 AUTENTIFICACIÓN

En seguida se describirá el protocolo de autenticación de **Kerberos**, es un proceso en el que diferentes elementos colaboran para conseguir identificar a un cliente que solicita un servicio ante un servidor que lo ofrece algún servicio en general; este proceso se realiza en tres grandes etapas que a continuación se describen. Abajo se muestran la notación utilizada, y en la figura 3.2 se ilustra gráficamente el protocolo [COULOURIS, et al., 1994], [19], [24], [25].

Notación:

- ***A*** Nombre del servidor de autentificación (AS).
- ***T*** Nombre del servidor de emisión de *tickets* (TGS).
- ***C*** Nombre del cliente que solicita un servicio en representación de un usuario.
- ***S*** Nombre del servidor de aplicación.
- K_C Clave secreta del cliente.
- K_S Clave secreta del servidor.
- K_T Clave secreta del servidor de emisión de *tickets*.
- K_{CT} Clave de sesión entre el cliente y el servidor de emisión de *tickets*.
- K_{CS} Clave de sesión entre el cliente y el servidor.

3.6.1 INICIO DE SESIÓN

Primero se describe el protocolo en el cual el cliente obtiene un *ticket* y una llave de sesión para poder acceder al *TGT*.

Inicialmente el cliente *C* necesita obtener las credenciales necesarias para acceder a otros servicios (donde el cliente proporciona su identidad al igual que un valor efímero N) y requiere que el servidor *A* le proporcione un ticket para poder comunicarse con *T*:

$$C \rightarrow A : C, T, N$$

Si el usuario es conocido, el servidor de autenticación retorna un mensaje que contiene una clave para la comunicación con *TGS* y un *timestamp* encriptado con la clave secreta del cliente, junto con un *ticket* para la comunicación con *TGS* cifrado con la clave secreta de este servidor:

$$A \rightarrow C : \{K_{CT}, N\}_{K_C}, \{ticket(C, T)\}_{K_T}$$

El cliente *C* intentará descifrar $\{K_{CT}, N\}K_C$, con la clave que el usuario proporciona, y si ésta es correcta podrá obtener K_{CT} y *N* : un cliente sólo podrá descifrar esta parte del mensaje si conoce su clave secreta, K_C (en este caso el *password*). Una vez obtenida K_{CT}, la clave para comunicar al cliente con el servidor de emisión de *tickets*, el cliente la guarda para una posterior comunicación con el *TGS* y borra la clave del usuario de memoria, ya que el *ticket* será suficiente para autenticar al cliente; este modelo consigue que el *password* nunca viaje por la red [COULOURIS, et al., 1994], [17], [19], [23], [24].

3.6.2 OBTENCIÓN DE *TICKETS*

El cliente ya tiene una clave de sesión para comunicarse con el servidor de emisión de *tickets* y el *ticket* necesario para hacerlo, encriptado con la clave secreta de este servidor (el cliente no puede descifrar este ticket). Cuando el cliente necesita acceder a un determinado servicio que se encuentre en el servidor S es necesario que disponga de un *ticket* para hacerlo, por lo que lo solicita uno al *TGS* enviándole un autenticador que el propio cliente genera, el *ticket* de *T* y el nombre del servicio al que desea acceder, *S*, y un indicador de tiempo:

$$C \rightarrow T : \{auth(C)\}_{K_{CT}}, \{ticket(C,T)\}_{K_T}, S, N$$

Cuando *TGS* recibe el *ticket* comprueba su validez y si todo es correcto genera una clave de sesión y retorna un mensaje que contiene dicha clave para comunicación con S y un *timestamp* cifrado con la clave de sesión del par *C* y *T*,

junto a un *ticket* para que el cliente *C* y el servidor *S* se puedan comunicar cifrado con la clave secreta del servidor:

$$T \rightarrow C : \{K_{CS}, N\}_{K_{CT}}, \{ticket(C, S)\}_{K_S}$$

El cliente *C* sólo podrá obtener K_{CS} si conoce la clave secreta, K_{CT} [COULOURIS, et al., 1994], [17], [29], [24], [25].

3.6.3 PETICIÓN DE SERVICIO

Tras obtener el ticket para comunicarse con S el cliente ya está preparado para solicitar el servicio; para ello presenta la credencial autenticada ante el servidor final, que es quien va a prestar el servicio. C se comporta de la misma forma que cuando solicitó un ticket a T: envía a S el autenticador recién generado, el ticket y una petición que puede ir cifrada si el servidor lo requiere, aunque no es necesario:

$$C \rightarrow S : \{auth(C)\}_{K_{cs}}, \{ticket(C, S)\}_{K_S}, petición, N$$

El servidor envía entonces al cliente la prueba autenticidad cifrada con la clave secreta de la sesión:

$$S \rightarrow C : \{N\}_{K_{cs}}$$

Sólo S pudo obtener K_{CS} y por tanto enviar este mensaje [COULOURIS, et al., 1994], [19], [24], [25].

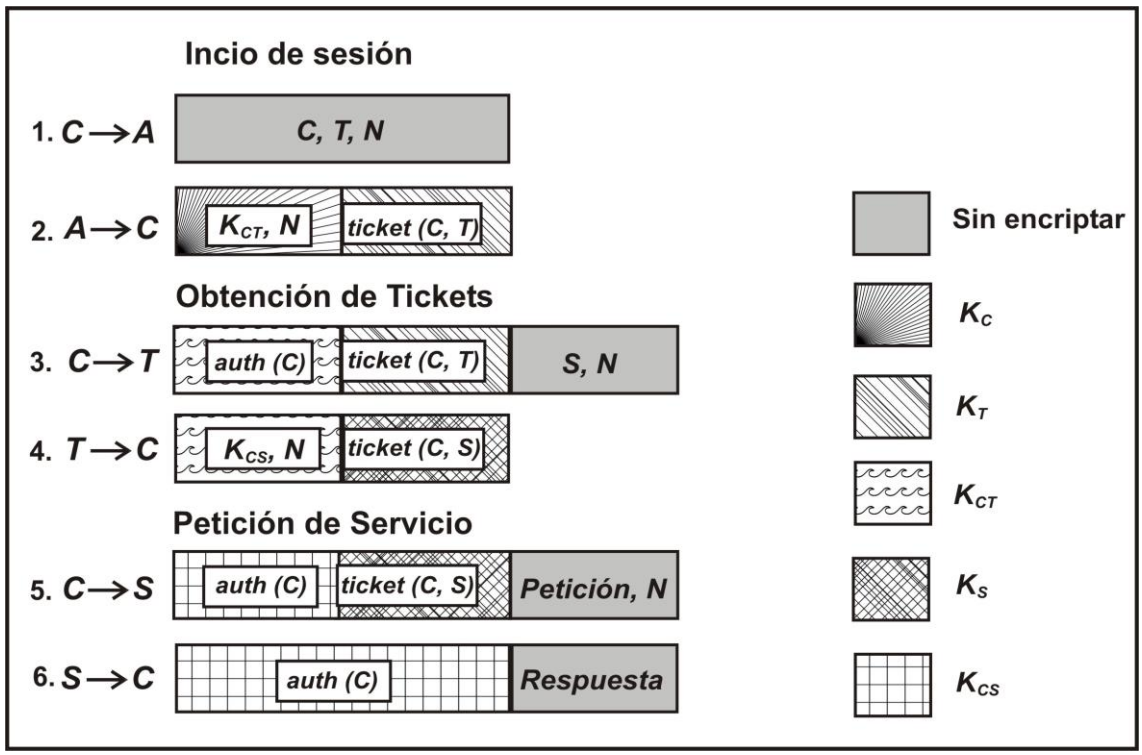

Figura 3.2 Diagrama de el Protocolo de Kerberos

3.7 VERSIÓN V4 DE KERBEROS

La versión V4 de **Kerberos** se basa en gran parte en los protocolos de *Needham y Schroeder*, con algunos cambios necesarios para complementar los requerimientos de los entornos de red para los que había sido originalmente diseñado. Entre los cambios mencionados se encuentran los siguientes:

- Uso de marcas temporales, puestos por *Denning* y *Sacco*.

- Adición de un servicio de emisión de *tickets* para dar soporte a las siguientes operaciones de autentificación sin necesidad que el usuario introduzca su contraseña constantemente.

- Una solución diferente en la autentificación entre reinos.

En la tabla 3.4 se puede observar la manera en que se formaliza los pasos del protocolo de la versión V4 de **Kerberos**.

Paso 1	C	$\rightarrow$ AS	:	U, TGS, T, L
Paso 2	AS	$\rightarrow$ C	:	$\{K, N, T_{c, tgs}\}K_U$
Paso 3	C	$\rightarrow$ TGS	:	$S, N, T_{c, tgs}, A_{c, tgs}$
Paso 4	TGS	$\rightarrow$ C	:	$\{T_{c, s}, K'\} K$
Paso 5	C	$\rightarrow$ S	:	$T_{c, s}, A_{c, s}$
Paso 6	S	$\rightarrow$ C	:	$\{T + 1\}K'$

Tabla 3.4 Protocolo de la Versión V4 de Kerberos

Notación:

C Nombre del cliente en representación de un usuario.

AS Servidor de autentificación.

TGS Servidor de concesión de *tickets*.

S Nombre del servidor de aplicación.

TGT *Ticket* de concesión de *tickets.*

T Marca temporal.

L Tiempo de vida de los *tickets.*

U Identificador del usuario.

K Clave de sesión.

En la anterior descripción del protocolo, le termino $T_{c, tgs} = \{U, C, TGS, T, L, K\} K_{TGS}$ y se ocupa para indicar un *TGT* emitido por el *AS* que es utilizado por el cliente *C* para autentificarse ante el *TGS* y así poder solicitar el correspondiente *ticket*. Analógicamente, el término $T_{c, s} = \{U, C, S, T', L', K'\} K_S$ se utiliza para indicar un *ticket* emitido por *TGS* que será usado por el cliente *C* para autentificarse con el servidor *S*. *T* y *T'* indica marcas temporales, *L* y *L'* indica tiempos de vida de los *tickets*. La marca temporal representa el numero de

segundos desde las 00:00:00 GMT (hora de *Greenwich*). Y el tiempo de vida máximo en la versión V4 de **Kerberos** es de alrededor de 21 horas [OPPLIEGER, 1998], [17], [19], [25], [26].

Los términos $A_{c,\ tgs} = \{C,\ T\}$ y $A_{c,\ s} = \{C,\ T'\}\ K'$ se usan para indicar los autentificadores de $T_{c,\ tgs}$ y de $T_{c,\ s}$, respectivamente. El autentificador ya ha sido definido como un registro de datos que contiene información que se puede demostrar que ha sido generada recientemente usando una clave de sesión conocida solamente por el cliente y por el servidor solicitado.

Los seis pasos que constituyen el protocolo de **Kerberos** se pueden agrupar en tres intercambios:

- El intercambio con *AS*, entre el cliente y el *AS* (paso 1 y 2).
- El intercambio con *TGT*, entre el cliente y el *TGT* (paso 3 y 4).
- El intercambio con S, entre el cliente y el servidor de aplicación (paso 5 y 6).

3.7.1 INTERCAMBIO CON AS

En la figura 3.3 muestra el intercambio con *AS* en la versión V4 de **Kerberos**. El cliente *C* utiliza u servicio de nombres para obtener una lista de TGT disponible en ese momento y selecciona el más cercano en términos de topología de red, posteriormente *C* envía un mensaje KRB_AS_REQ (solicitud al servidor de autentificación de **Kerberos**) al AS (paso 1 de la tabla 3.3). El mensaje contiene el identificador del usuario *U* el identificador del *TGS,* una marca temporal *T* y el tiempo de vida deseado *L*, para el *TGT*.

Una vez recibido el mensaje KRB_AS_REQ, el AS busca y extrae la clave secreta asociada con *U* y el *TGT*. El *AS* selecciona genera de forma aleatoria una nueva clave de sesión, *K*, y devuelve el mensaje KRB_AS_REQ (respuesta del servidor de autentificación de **Kerberos**) a *C* (paso 2 de la tabla 3.3). El mensaje contiene K, N, y el $T_{c,\ tgs} = \{U,\ C,\ TGT,\ T,\ L,\ K\}\ K_{TGS}$, encriptado con K_U [OPPLIEGER, 1998], [19], [25], [26].

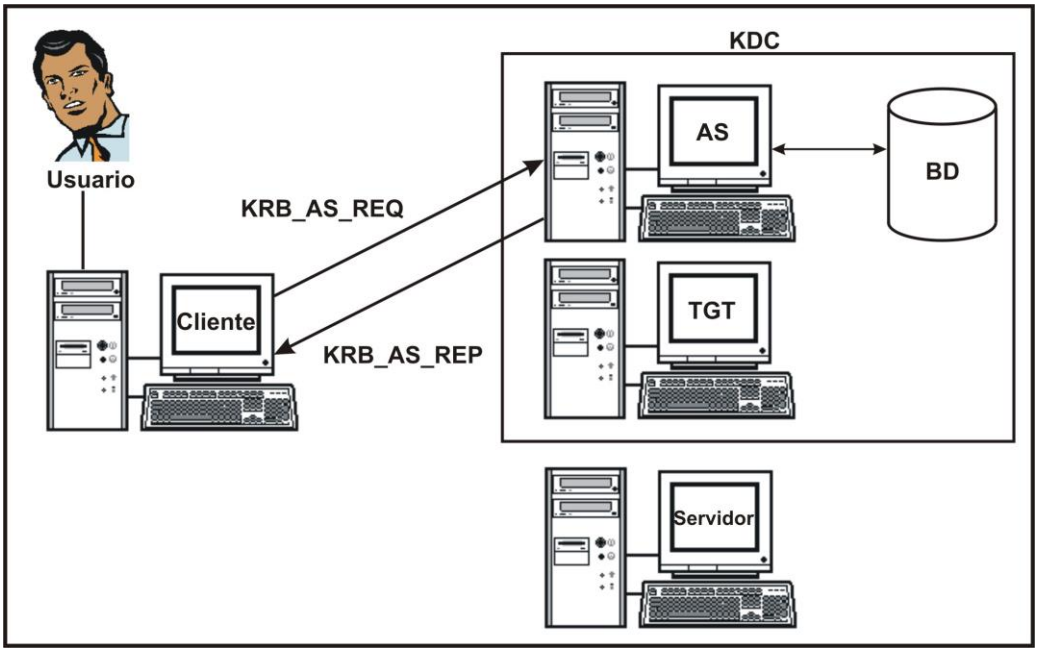

Figura 3.3 Intercambio entre el cliente y AS

Después de recibir el mensaje KRB_AS_REP se le solicita a *U* que introduzca su contraseña (*password*). En la versión V4 de **Kerberos** no solicita al usuario dicha contraseña hasta que *C* ha recibido el mensaje KRB_AS_REP, debido esta versión es muy formal en seguir el principio de seguridad en el que *C* utilice la contraseña del usuario el tiempo mínimo posible. Pero la espera de varios segundos hasta recibir el mensaje KRB_AS_REP, antes de pregunta al usuario su contraseña no mejora propiamente la seguridad, incluso en la versión V5 de **Kerberos** el usuario proporciona su contraseña antes que *C* envíe el mensaje KRB_AS_REQ. La razón que tuvieron los diseñadores de la versión V5 para mitificar este paso fue que dicha versión exige que *C* pruebe que conoce la contraseña del usuario antes que el *AS* envíe el mensaje KRB_AS_REP, y esto dificulta la posibilidad de que alguien obtenga la información con la cual se podría lanzar un ataque de búsqueda de contraseña.

En la Versión V4 si *U* introduce correctamente su contraseña; *pwd*, *C* puede usar una fusión *Hash* unidireccional *h* para calcular la clave secreta $K_U = h(pwd_U)$. Con

esta clave, *C* puede desencriptar *{K, N, T,* $T_{c,\ tgs}$ *}*K_U y extraer en secuencia *K, N,* y $T_{c,\ tgs}$. Si *C* tiene éxito, tendrá un *TGT* que utilizara para solicitar *tickets* del *TGS*.
En un *TGT*, el campo de tiempo de vida se utiliza como tiempo de expiración de la contraseña. Al limitar el tiempo de vida también se limita el daño que se puede causar en el caso que el *TGT* se vea comprometido. En **Kerberos** generalmente no se pude revocar un *TGT* una vez emitido. Por lo tanto, al limitar el tiempo de vida, implícitamente se establece un momento a partir del cual el *TGT* es obsoleto [OPPLIEGER, 1998], [19], [25].

3.7.2 INTERCAMBIO CON TGS

Los formatos de los mensajes en el intercambio con el *TGS* son prácticamente similares al intercambio con el *AS* como se muestra en la figura 3.4. La diferencia es que el encripado y desencriptado se realiza con la clave de sesión *K* que *C* comparte ahora con el TGS.

Antes de comenzar el intercambio, *C* debe determinar en qué reino esta registrado el servidor al que le solicitara el *ticket.* Si *C* no posee un *TGT* para ese reino, tendrá que conseguir uno. Primero se le solicita al servidor local de **Kerberos** un *TGT* para el reino deseado (usando recursivamente el mensaje KRB_TGS_REQ). El servidor de **Kerberos** puede entonces devolver el *TGT* para el reino deseado, en consecuencia C puede proceder. Aun que tan bien puede suceder que **Kerberos** devuelva un TGT para un reino más cercano al deseado. Si no devuelve ningún *ticket*, se solicitud deberá repetirse utilizando un servidor **Kerberos** del reino de jerarquía superior. Una vez que *C* obtiene el TGT del reino apropiado, determina que servidor de **Kerberos** se ocupa de ese reino y lo contacta. La lista puede obtenerse mediante un archivo de configuración o mediante el correspondiente servicio de red [OPPLIEGER, 1998].
C envía el mensaje KRB_TGS_REQ (solicitud al servidor de *tickets* de **Kerberos**) al *TGS* (paso 3 de la tabla 3.4). El mensaje contiene *S, N,* $T_{c,\ tgs}$ y un autenticador $A_{c,\ tgs}$ *= {C, T}* K, donde *T* es una marca temporal. Adviértase que $T_{c,\ tgs}$ puede

tener un tiempo de vida relativamente prolongado, y podría ser rastreado y duplicado. Por lo tanto, el objetivo del autenticador es demostrar que C tiene la clave secreta, y así evitar este tipo de ataque.

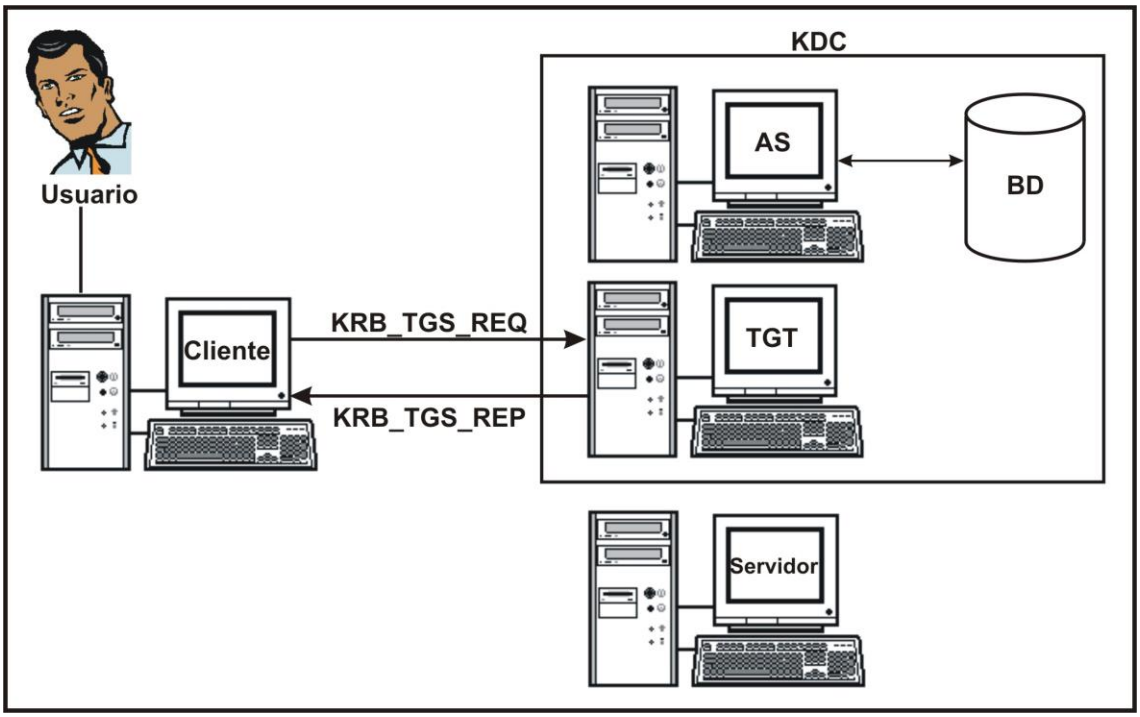

Figura 3.4 Intercambio entre el cliente y TGS

El mensaje KRB_TGS_REQ se procesa de forma parecida al mensaje de KRB_AS_REQ, pero se realizan algunas comprobaciones adicionales. Posteriormente el *TGS* devuelve el mensaje KRB_TGS_REP (respuesta del servidor de emisión de *tickets* de **Kerberos**) a *C* (paso 4 de la tabla 3.4). Dicho mensaje de respuesta tiene el mismo formato que le mensaje KRB_AS_REP, sin embargo incluye $T_{c,s}$ para el servidor solicitado *S*, y una nueva clave de sesión *K'* ambos cifrados con *K*. Cuan C recibe el mensaje lo procesa de la misma forma que la respuesta del AS descrito anteriormente. La principal diferencia es que la parte del texto encriptado de la respuesta, debe se desencriptada con la clave de sesión compartida con el *TGT* [OPPLIEGER, 1998], [19], [25].

3.7.3 INTERCAMBIO CON EL SERVIDOR DE APLICACIÓN.

Como se muestra en la figura 3.5 el intercambio en servidor de aplicación S, dicho intercambio sirve para autentificar a un cliente ante un servidor, o e para la autentificación mutua entre cliente y servidor. El cliente *C* debe haber obtenido las credenciales necesarias para el servidor requerido, mediante los intercambios con *AS* y *TGS* [OPPLIEGER, 1998].

C envía el mensaje KRB_S_REQ (solicitud de aplicación) al servidor de aplicación S (paso 5 de la tabal 3.5). El mensaje contiene $T_{c,\ s}$, $A_{c,\ s}$ = *{C, T'}K'*. La autentificación se basa en la cuenta de tiempo del servidor, el *ticket* $T_{c,\ s}$ y el autenticador $A_{c,\ s}$.

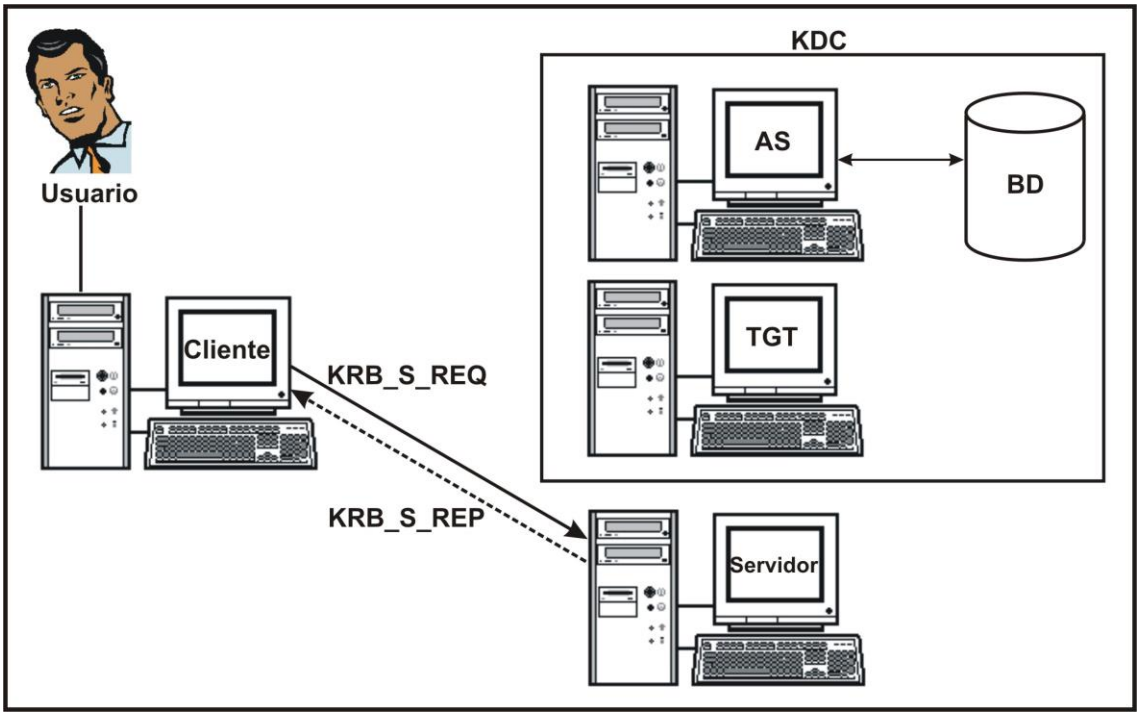

Figura 3.5 Intercambio entre el cliente y S

Si no se producen errores, y en el caso que se requiera la autentificación mutua, S debe enviar el mensaje KRB_S_REP (respuesta de aplicación) a C (paso 6 de la tabal 3.5). Este mensaje se encripta con la clave de sesión K' , compartida por C y

S. Debido a que esta clave se encontraba en el ticket encriptada con la clave secreta del servidor, la posesión de dicha clave, prueba que S es el servidor deseado. De una manera concreta, S debe incrementar las marcas temporales incluidas en el autentificador del mensaje KRB_S_REQ y volver a cifrarla con K' [OPPLIEGER, 1998], [19], [25].

3.7.4 CONFIDENCIALIDAD DE DATOS Y SERVICIOS DE INTEGRIDAD

Kerberos proporciona servicios de autentificación, sin embargo, un resultado adicional del servicio de autentificación de **Kerberos** es el intercambio de una clave de sesión *K'* compartida entre el cliente y el servidor. Esta clave puede ser empleada en lo sucesivo por la aplicación para proteger la integridad y la confidencialidad de las comunicaciones. En realidad el sistema **Kerberos** proporciona dos tipos de mensajes [OPPLIEGER, 1998]:

- El mensaje KRB_PRIV, para proteger la confidencialidad de las comunicaciones, utiliza una suma de comprobación criptográfica.
- El mensaje KRB_SAFE, para proteger la integridad de las comunicaciones, utiliza la encriptación.

No obstante de la existencia de estos tipos de mensaje proporcionados por el sistema las aplicaciones son libres de utilizar cualquier método que se ajuste mejor a los datos concretos que se vayan a transmitir [OPPLIEGER, 1998].

3.8 VERSIÓN V5 DE KERBEROS

En esta sección se describe la versión V5 de **Kerberos** en general, y se describen las principales diferencias con la versión 4. La versión V5 incorpora los siguientes cambios [OPPLIEGER, 1998], [19]:

- **Identificadores de los principales**: En la versión V5, un identificador de un servidor consta de dos partes; el reino y el resto. El reino, el nombre de este se encuentra separado para facilitarlos la ejecución de los procesos que involucren varios reinos y comprobaciones de acceso dependientes del reino. El resto es una secuencia de tantos componentes como sea necesario para nombrara al servidor**.**

- **Uso de cifrado:** Para mejorar la modularidad de **Kerberos**, el encriptado está separado en diferentes módulos de software que pueden ser sustituidos o eliminados a criterio de los programadores. Cuando un mensaje utiliza el encriptado, dicho texto encriptado se marca con un identificador para que el receptor pueda saber el algoritmo que debe utilizar para desencriptar el mensaje. Cada algoritmo de encriptación tiene como objetivo proporcionar la protección necesaria para mantener la integridad del texto sin encriptar, de modo que el receptor pueda comprobar que el texto encriptado no ha sido alterado durante su recorrido. Si el algoritmo de encriptado no tiene dicha propiedad, se puede incluir una suma de comprobación al texto antes de cifrarlo. Es decir, con el modo CBC estándar de DES, añadiendo al texto sin encriptar una suma de comprobación.

- **Direcciones de red:** En la versión V5 de **Kerberos**, los mensajes que se envían el cliente a los distintos servidores de **Kerberos** contienen direcciones de red, se identifican análogamente con un tipo y cierta longitud de campo, de manera que el receptor pueda recibirlos adecuadamente. Si un sistema admite múltiples protocolos de red o tiene múltiples direcciones de un solo tipo en un *ticket* se puede proporcionar se pueden proporcionar todos los tipos y direcciones.

- **Codificación de mensajes:** En la versión V5 se ha ampliado el formato del *ticket* para dar cabida a las modificaciones que requería la versión V4. Está

dividido en dos parte, una esta encriptada y la otra no. El nombre del servidor en el *ticket* se encuentra sin encriptar, ya que un servidor con múltiples con identidades, por ejemplo, un TGS empleado en varios reinos, puede requerir el nombre para seleccionar una clave para seleccionar una clave con la cual descifraría todo el *ticket*. Realmente, el nombre del servidor, es solo información de contabilidad y debido a esto no es necesaria su protección para una autentificación segura, todo lo demás esta encriptado. Además de éste cambio, el tiempo de validez del ticket esta expresado como un tiempo de emisión y un tiempo de expiración (en lugar de ser un campo de tiempo de validez). Esto permite emitir tiempos de validez prácticamente sin límite.

- **Soporte para operación entre reinos:** En esta versión los reinos pueden cooperar a través de una jerarquía que se basa en el nombre del reino. Un reino fuente puede operar con un reino destino si comparten directamente una clave entre reinos o si comparte una clave con un reino intermedio, que a su vez, pueda operar con el reino destino. Cada nodo intercambia con su nodo padre y con cada hijo una pareja diferente de claves. Estas claves se utilizan en un criptosistema con clave secreta para obtener los *tickets* de cada reino sucesivo a través de la vía de la autentificación.

Además de estos cambios, la versión V5 tiene nuevas, por ejemplo, el campo de indicadores mencionados anteriormente permiten mayor flexibilidad en el uso de *tickets* que en la versión anterior. Cada *ticket* emitido por el *AS* que realiza el intercambio inicial es etiquetado como tal. Esto permite que los servidores, por ejemplo, los servidores de intercambio de contraseñas, exijan que el cliente presente su *ticket* obtenido por uso directo de la clave secreta del cliente, en lugar de utilizar un *ticket* obtenido mediante un *TGT*. Esto evita que un posible atacante pueda entrar en una estación activa, en la que no se esté operando en ese momento y cambie la contraseña de otro usuario.

Los *tickets* se pueden emitir con la característica de renovables, asociándoles dos tipos de caducidad o expiración: uno de ellos en un futuro cercano y el otro posterior. El *ticket* expira en el primero de esos tiempos en la forma usual. Pero si antes de la fecha de caducidad se le presenta al *TGS* una solicitud de renovación, se puede regresar un ticket de reemplazo que es válido durante un intervalo de tiempo adicional. Pero el *TGS* no renovara el ticket más del segundo instante de expiración indicado en el mismo. Este procedimiento tienen la ventaja que aunque las credenciales se puedan utilizar durante periodos prolongados de tiempo, el *TGS* puede rechazar la renovación de los tickets marcados como robados, evitando así su utilización continua. Existe un mecanismo similar para ayudar a la autentificación durante el procesamiento por lotes. Si se emite un *ticket* con fecha posterior, no será válido hasta que llegue esa fecha y sea reemplazado con un ticket valido. El cliente valida el *ticket* mostrándoselo al *TGS* de la misma manera que ya se ha descrito en los *tickets* renovables.

Kerberos se encarga esencialmente de la autentificación, no se ocupa de los servicios relacionados de la seguridad, como las autorizaciones. Para poder implementar los servicios de seguridad, la versión V5 proporciona un mecanismo para la trasmisión de información de autorización a prueba de alteraciones como parte de un *ticket*. Esta información puede restringir el uso del *ticket*, la codificación de la correspondiente restricción no es asunto del sistema **Kerberos**, esta se encontrara definida en los mecanismos de autorización de los que se utilicen. Las restricciones se codifican en el campo de datos de autorización del *ticket*. De manera que cuando se solicita un *ticket*, las restricciones se establecen y se envían la *TGS*, donde se insertan en el *ticket*, se encripta y así se protege contra alteraciones. En la forma más general de esta versión, el cliente le pide al *KDC* incluya estos datos en un nuevo *ticket*, el *TGS* no elimina del *TGT* ningún dato sobre autorización, sino que los copia en cada nuevo ticket y agrega los datos adicionales sobre la autorización que se requiera. Después del desencriptado del *ticket*, los datos de autorización estarán a disposición del servidor de aplicación. Aun que **Kerberos** no realiza ninguna interpretación de los datos espera que el servidor de aplicación utilice los datos de autorización para

restringir el acceso del cliente al recurso especificado en el *ticket*. Entre otros usos, el campo de autorización puede ser también utilizado por un *ticket* de sustitución para crear una capacidad. El cliente que solicita el ticket de sustitución a un *TGS* establece las restricciones en el campo de datos de autorización, posteriormente transmite de forma segura la clave de sustitución y de sesión, que utiliza para obtener el servicio limitado del servidor de aplicación.

En un esfuerzo para dificultar el robo de contraseñas, la versión V5 de **Kerberos** ofrece un campo de datos de pre-autentificación para incluir alternativas a las contraseñas. En el intercambio inicial de *AS* el campo de datos de pre-autentificación se puede utilizar para modificar la clave con la que se va a enviar la respuesta. Con esto se consigue que si la contraseña es robada, carezca de utilidad, ya que para desencriptar una respuesta.

En el manejo de la confidencialidad de los datos y el servicio de integridad, los mensajes KRB_SAFE y KRB_PRIV en la versión V4 incluyen información de control, una marca temporal y la dirección de red del remitente. Con la versión V5, las aplicaciones pueden escoger entre usar la marca temporal o un numero de secuencia, de esta manera el receptor debe verificar que los mensaje llegue en orden apropiado sin saltos entre ellos[OPPLIEGER, 1998], [19], [27].

En la tabla 3.5 se muestra el protocolo que sigue la versión V5 de **Kerberos**.

Paso 1	$C \rightarrow AS : U, TGS, L_1, N_1$
Paso 2	$AS \rightarrow C : U, T_{c,tgs}\{TGS, K, T_{inicio}, T_{fin}, N_1\}K_U$
Paso 3	$C \rightarrow TGS : S, L_2, N_2, T_{c,tgs}, A_{c,tgs}$
Paso 4	$TGS \rightarrow C : U, T_{c,s}, \{S, K', T'_{inicio}, T_{fin}, N_2\} K$
Paso 5	$C \rightarrow S : T_{c,s}, A_{c,s}$
Paso 6	$S \rightarrow C : \{T'\}K'$

Tabla 3.5 Protocolo de la Versión V5 de Kerberos

De la misma forma que en la versión V4 $T_{c,\ tgs}$, $T_{c,\ s}$ se utiliza para indicar, respectivamente, un *TGS* y un ticket. $A_{c,\ tgs}$ *y* $A_{c,\ s}$ se utilizan para indicar los autentificadores correspondientes.

Notación:

C	Nombre del cliente en representación de un usuario.
AS	Servidor de autentificación.
TGS	Servidor de concesión de *tickets*.
S	Nombre del servidor de aplicación.
N	Valor temporal.
T	Marca temporal
L	Tiempo de vida.
U	Identificador del usuario.
K	Clave de sesión.

En el paso 1 del tabla 3.5, *C* elige un *TGS* y envía el correspondiente mensaje KRB_AS_REQ al *AS* de **Kerberos**. El mensaje contiene el identificador de usuario *U*, el identificador del *TGS* elegido, un tiempo de vida solicitado para el *TGT*, L_1 y un valor temporal N_1. Además el mensaje puede establecer nuevas opciones como:

- Si se realizara pre-autentificación.
- Si el ticket elegido será renovable, sustituible o reenviable.
- Si su validez está retrasada o se permitirá la validez retrasada de los *tickets*.
- Si se aceptará un ticket renovable en lugar de un ticket no renovable en el caso que la fecha de expiración del ticket no pueda satisfacerlo utilizando un ticket no renovable.

Después de recibir le mensaje KRB_AS_REQ el AS busca y extrae las claves secretas de *U* y del *TGT*. Si se necesita, el *AS* pre-autentifica la solicitud, y si falla, se le regresa a *C* el respectivo mensaje de error. De lo contrario, el *AS* selecciona aleatoriamente una nueva clave de sesión K, y regresa a *C* el mensaje KRB_AS_REP en el paso 2 de la tabla 3.5. El mensaje contiene *U*, un *TGT*, un $T_{c, tgs} = \{U, C, TGS, K, T_{inicio}, T_{fin}\} K_{tgs}$ y $\{TGS, K, T_{inicio}, T_{fin}, N_1\}K_U$. Los tiempos de emisión y de expiración del *TGT*, T_{inicio} y T_{fin}, son establecidos de acuerdo con la política de seguridad del reino, de modo que es probable que se ajuste al tiempo de vida L_1 establecido en el mensaje KRB_AS_REQ.
Inmediatamente después de recibir el mensaje KRB_AS_REP, C utiliza una función hash unidireccional *h* a la contraseña proporcionada por el usuario pwd_U, para calcular la clave maestra $K_U = h(pwd_U)$. Con esta clave *C* puede desencriptar $\{TGS, K, T_{inicio}, T_{fin}\} K_U$ y así poder extraer $TGS, K, T_{inicio}, T_{fin}$. Ahora *C* tiene un TGT válido desde T_{inicio} hasta T_{fin}, y puede usar *TGT* para solicitar tickets al *TGS*.

El intercambio con *TGS* inicia cuando *C* requiere un ticket de un servidor, renovar o validar un ticket existente o incluso obtener un ticket de intercambio. Por lo menos en el primer caso, el cliente debe haber obtenido en ese instante un *TGT* por medio del intercambio con *AS*. En el paso 3 de la tabla 3.5, *C* manda al *TGS* un mensaje KRB_TGS_REQ, este mensaje posee información con la autentificación del cliente y una solicitud de credenciales. La información de autentificación contiene el $T_{c, tgs}$ del *TGT* y su autentificador correspondiente $A_{c, tgs}\{C; T\}K$ que *C* genera con una marca temporal *T*, usando la clave de sesión *K*. En el paso 4 de la tabla 3.5 el *TGS* devuelve un mensaje KRB_TGS_REP que contiene un ticket $T_{c, s} = \{U, C, S, K', T'_{inicio}, T'_{fin}\}K_S$ para el servidor solicitado, así como $\{S, K', T'_{inicio}, T'_{fin}\}K$.

El intercambio con *S* es utilizado nuevamente para autentificar a un cliente ante un servidor o para la autentificación mutua entre cliente y servidor. En el paso 5 de la tabla 3.5 *C* envía al servidor de aplicación *S* el mensaje KRB_S_REQ que contiene el ticket $T_{c, s}$ y el correspondiente autentificador $A_{c, s}\{C, T'\}K'$. La

autentificación se basa en la hora del día actual que tiene el servidor, el autentificador y el ticket. Si no existen errores y se requiere autentificación mutua, *S* devuelve un mensaje KRB_S_REP en el paso 6 de la tabla 3.5. Dicho mensaje contiene la marca temporal *T'*, encriptada con la clave de sesión *K'* que *S* y *C* comparten en ese instante [OPPLIEGER, 1998], [19], [27].

3.9 AUTENTIFICACIÓN ENTRE REINOS

Un entorno de red puede estas constituido por varias organizaciones como empresas en competencia mutua, agencias gubernamentales, instituciones financieras o universidades. En algún entorno de los anteriores, sería difícil encontrar una entidad de confianza para que ejecute un *KDC* de **Kerberos**. El mayor problema es que con el modelo **Kerberos** el encargado de ejecutar y gestionar el *KDC* puede tener acceso a cualquier clave maestra de usuario, en consecuencia puede entrar a todo lo que el usuario tenga acceso. Inclusive si hubiese una entidad en la que todos los usuarios confiaran, no es suficiente. Todos deben confiar en los controles físicos para cada réplica del *KDC* y podría haber réplicas dispersas en una amplia zona geográfica por conveniencia. El compromiso de cualquier réplica, sin importar lo oculta que se encuentre, comprometería todas las claves de los principales. Además, es una entidad de lata fiabilidad y confianza, estaría siempre demasiado ocupada, ya que tendría que procesar todas las ejecuciones de los usuarios y de los servicios que entran y salen del entorno de red.

Debido a esto, los entornos de red se dividen en reinos, cada reino tiene y ejecuta su propio *KDC*. En un reino puede haber varios *KDC*, pero serian equivalentes, tendrían la misma clave maestra y las mismas entradas en la base de datos. Pero dos *KDC* de diferentes reinos tendrían claves maestra diferentes, y las bases de datos de claves maestras de los principales completamente diferentes, ya que serán responsables de conjuntos de principales diferentes.

Kerberos se diseño para traspasar las barreras entre organizaciones entre organizaciones. En principio el cliente de un reino se puede autentificar ante un servidor de otros reinos. En **Kerberos** a este procedimiento se le llama o denomina **autentificación entre reinos**.

Una clave entre reinos, es una clave compartida entre los *KDC* de dos reinos de **Kerberos**. Al establecerse la clave entre reinos, los administradores de los reinos, permiten que un cliente autentificado en un reino pueda usar remotamente su autentificación. El intercambio de claves entre reinos registra al *TGS* de cada reino como un principal del otro reino. De manera que los clientes pueden obtener un *TGT* para el reino remoto desde el *TGS* del reino local. De manera que cuando se utilizan esos *TGT*, el *TGS* remoto utiliza la clave entre reinos para desencriptar el *TGT*, y comprobar así que ha sido enviado por el *TGS* del cliente. Los *tickets* enviados por el TGS remoto indicaran a un servicio que el cliente ha sido autentificado en otro reino.

Solo se puede comunicar un reino con otro si ambos comparten una clave entre reinos, o también si el reino local comparte un clave entre reinos con un reino intermedio que se comunica con el reino remoto. En términos de **Kerberos**, se define vía de autentificación como la secuencia de reinos intermedios que se cruzan en el proceso de autentificación cuando un reino se comunica con otro.

En la Versión V4 es necesario que los AS se registre en cada reino remoto con él se ha solicitado una autentificación entre reinos, de este manera, si *n* es el numero de reino, la interconexión completa requiere n^2 claves. En cambio en la versión V5 admite la autentificación miltipaso entre reinos, de manera que las claves se pueden compartir jerárquicamente. Cada reino comparte claves con su reino padre y sus reinos hijos. Si dos reinos no comparten una clave entre reinos directamente, la organización jerárquica permite que se establezca una vía de autentificación. Y si no se utiliza una organización jerárquica, se puede necesitar consultar una base de datos para construir una vía de autentificación entre reinos.

Aunque los reinos se organizan por lo regular de forma jerárquica, los reinos intermedios se pueden omitir para realizar la autentificación entre reinos mediante vía alternativas, con esto se puede conseguir una comunicación más eficiente entre dos reinos. Es importante que el solicitante del servicio sepa que reinos se cruzan en el proceso de autentificación, para facilitar esto existe un campo en el *ticket* que contiene la secuencia de los reinos atravesados, excluyendo los reinos fuente y destino [OPPLIEGER, 1998].

3.10 PROBLEMAS DE KERBEROS

El uso de **Kerberos** mejora notablemente la seguridad de las aplicaciones de red, haciendo que le sea más difícil a alguien suplantar a otro usuario, y aunque se trate de un sistema robusto, el sistema de autentificación de **Kerberos** no es una solución definitiva para todos los problemas de la seguridad en las redes ya que también cuenta con sus propios problemas, tanto de seguridad como de implementación.

Uno de los principales problemas de **Kerberos** es que cualquier programa que lo utilice ha de ser modificado para poder funcionar correctamente, siguiendo un proceso denominado `kerberización'. Esto implica obviamente que se ha de disponer del código fuente de cada aplicación que se desee kerberizar, y también supone una inversión de tiempo considerable para algunas aplicaciones más o menos complejas que no todas las organizaciones se pueden permitir. El problema anterior es simplemente de implementación; no afecta en nada a la seguridad del sistema. Un problema que sí está relacionado con la seguridad de **Kerberos** es la gran centralización que presenta el sistema, para un óptimo funcionamiento se ha de disponer en todo momento del servidor **Kerberos**, de manera que si la máquina que lo alberga falla, la red dejara de funcionar. Innegablemente esto es una contradicción con lo que se especifica en la teoría de sistemas distribuidos, donde se subraya el uso de la distribución para mantener la disponibilidad del sistema, intentado que si un equipo falla, el resto pueda seguir

funcionando, si no a pleno rendimiento, al menos correctamente. Por si esto no fuera suficiente, la centralización de **Kerberos** reside en el hecho de que casi toda la seguridad reside en el servidor que mantiene la base de datos de claves, de manera que si éste se ve comprometido, la red entera se está amenazada.

Un potencial problema de seguridad es el uso de *timestamps*, esto obliga a que todas las máquinas que ejecutan servicios autentificados mantengan sus relojes mínimamente sincronizados (con desfases máximos de pocos minutos), con todo lo que esto implica. Además ese tiempo global ha de ser accesible a todas las estaciones; aunque en el diseño no se asume que todas mantengan la hora exacta, sí que se les obliga a mantenerse dentro de los márgenes si desean solicitar *tickets*, para lo que se necesitan servidores de tiempo con los que los clientes puedan sincronizar periódicamente sus relojes, por ejemplo cada vez que arrancan.

Todos estos problemas, y algunos más que se han ido solucionando en diferentes versiones del sistema. Algunos críticos como *Bellovin* y *Merritt* describieron algunas deficiencias y limitaciones en versión V4 de **Kerberos**. A continuación se describirán los resultados de estos estudios, primero las deficiencias de entorno y posteriormente las deficiencias técnicas.

En las deficiencias de entorno se encuentran los siguientes puntos:

- **Dependencia del sistema de cifrado:** la versión V4 de **Kerberos** utiliza el DES (*Data Encryption Standar*) para encriptar los mensajes. Pero la exportación del DES fuera de los EEUU está restringida por su gobierno, haciendo difícil su uso verdaderamente amplio de la versión V4 de **Kerberos**, aunque esta limitación se aplica al sistema **Kerberos** en cualquier versión.

- **Dependencia del protocolo de Internet:** la Versión 4 requiere el uso de direcciones del protocolo de Internet (IP), lo que lo hace inadecuado para algunos entornos.

- **Tiempo de vida del *ticket:*** En esta versión, el máximo tiempo de vida es ligeramente superior a las 21 horas. Este límite es una de las principales deficiencias, ya que imposibilita la asignación de *tickets* a tareas por lotes de larga duración.

- **Reenvió de la autentificación:** en la versión V4 no realiza ninguna previsión para permitir que los *tickets* enviados a los clientes en un sistema, se puedan reenviar a otro sistema, o que puedan ser utilizados por otro cliente. Esta opción podría ser interesante si un servidor intermedio necesita acceder a algún recurso con los derechos del cliente, o si un usuario entra en otro sistema de la red y desea realizar alguna actividad ahí con los mismos privilegios y la misma autentificación que tenía en el sistema original.

- **Nomenclatura de principales:** Los principales se nombran ocupando tres componentes: nombre, realización y reino. Cada uno de ellos puede tener una longitud de hasta 40 caracteres, pero estos tamaños resultaron ser cortos en algunas aplicaciones y entornos de aplicación.

- **Autenticación entre reinos:** la Versión V4 permite que los reinos mientras compartan una clave secreta. Un determinado cliente puede obtener tickets para servicios en otros reinos, primero consiguiendo un *TGT* para el reino externo a través de su *KDC* local y utilizando después el *TGT* para obtener el *ticket* en el servidor de aplicación del reino externo. Este intercambio de *tickets* por parejas hace que las solicitudes de tickets entre reinos sean fáciles de realizar, pero la interconexión de *n* reinos requiere n^2 claves, incluso cuando hay pocos reinos interconectados, la asignación y gestión de este número de claves entre reinos es una tarea costosa.

Además de las deficiencias de entorno, existen deficiencias técnicas en la Versión V4. Estas deficiencias son las siguientes:

- **Doble cifrado:** En la versión V4 el TGT emitido por AS están encriptados dos veces, y solo una vez cuando se envía al TGS. El segundo cifrado no es necesario y computacionalmente no es económico.

- **Cifrado PCBC:** en la versión V4 de **Kerberos** utiliza una versión no estandarizada del modo CBC, denominada encadenamiento de bloque cifrado y plano (sin encriptar) (PCBC – *plain and cipher block chaining*). En el modelo PCBC, un bloque sin encrifrar P_{i+1}, antes de ser encriptado es combinado mediante una operación de OR exclusivo en el texto sin encriptar P_i y con el texto encriptado C_i. Este modo posee la propiedad del que al ser modificado un bloque de texto encriptado C_i se modifican todos los bloques de texto desencriptado desde el P_i en adelante. La versión V4 coloca unos datos reconocibles al final del mensaje que se va a encriptar, de forma que se pueda reconocer si el bloque final se desencripta correctamente, entonces los datos no han sido alterados entre su transmisión por la fuente y la recepción en el destino. Sin embargo, el uso de DES en modo PCBC y la comprobación de los contenidos sólo en el bloque final desencriptados no garantiza que **Kerberos** sea capaz de detectar todos las alteraciones que pueden contener los mensajes.

- **Autentificación de mensajes:** De manera similar, en lugar de utilizar DES en modo CBC para la protección de la integridad, esta versión utiliza un algoritmo no estándar de suma de comprobación. La idea es calcular una suma de comprobación utilizando la clave de sesión concatenada al mensaje y transmitir al receptor tanto el mensaje como la suma de comprobación que se utiliza en la versión V4 no se encuentra documentado ni publicado, sin embargo, nadie ha conseguido hasta ahora demostrar la posibilidad de romper la seguridad del algoritmo, esto puede ser porque no

ha sido ampliamente utilizado en aplicaciones en las que mereciera la pena el esfuerzo de romper la seguridad del esquema. De manera general, el hecho de que la seguridad de que un algoritmo no se haya visto comprometida hasta el momento no es más que una evidencia débil de su seguridad.

- **Autentificadores y detección de réplicas:** V4 utiliza una marca temporal encriptada para verificar la antigüedad de los mensajes que se envían a los servidores y para evitar que un intruso pueda realizar con éxito un ataque de réplica. Sin un autentificador que contiene una marca temporal está caducado o está siendo replicado, el servidor de aplicaciones rechaza la autentificación, Pero, el mantenimiento de una lista de autentificadores caducados que han sido utilizados es difícil de realizar adecuadamente.

- **Claves de sesión:** la clave de sesión debe ser usada por el cliente y el servidor para proteger mensajes transmitidos durante una sesión, de esta manera, como el mismo *ticket* debe ser usado repetidas veces para obtener el servicio desde un servidor particular, existe el riesgo que un intruso repita el mensaje desde una sesión vieja al cliente o al servidor.

Algunas de las deficiencias de entorno y técnicas de V4 condujeron al diseño de la versión V5 de **Kerberos** mientras que otras no se han contemplado hasta el momento [OPPLIEGER, 1998], [BELLOVIN; MERRITT, 1991], [17], [23], [25], [28].

CONCLUSIONES

Como se ha podido constatar a lo largo de la historia, la criptografía es tan antigua como el hombre mismo y la necesidad de resguardar la información que comparte o envía para entidades definidas, las cuales pueden ser interceptadas por una tercera entidad. De esta forma ha evolucionado desde el citalo utilizado por Julio Cesar hasta nuestra actualidad como el caso de del gobierno de Suiza al transmitir los resultados de las elecciones federales del año 2007 vía internet donde emplearon una criptografía cuántica que se cree inviolable.

En la búsqueda de resguardar de una manera confiable la información se crea el protocolo Kerberos como sistema de autentificación que al integrar el algoritmo criptográfico DES lo convirtió en un sistemas robusto en contra de los accesos no autorizados y permitía conservar la integridad de los datos enviados en la red, dando como resultado una evolución creciente sobre la seguridad en redes distribuidas. En el desarrollo de Kerberos se perseguía un doble objetivo; que al viajar la información atreves de la red fuera interceptada por usuarios no autorizados y centralizar la autentificación, con lo cual se logro tener una base de datos única de todos los usuarios de la red.

Lo único que hace vulnerable al sistema de autentificación es las limitaciones del algoritmo criptográfico, ya que pueden ser rotos en algún momento, sin embargo el protocolo como tal es muy confiable, debido a ello muchos sistemas operativos lo utilizan como protocolo de autentificación tal es el caso de Windows 2000, Windows XP y Windows Server 2003 usan una variante de Kerberos. Así como Mac OS X de Apple también usa Kerberos tanto en sus versiones de cliente y de servidor, OS/400 de IBM y Red Hat Linux por mencionar algunos.

Por lo que podemos decir que no se ha escrito la última pagina en cuanto a criptografía y sistemas de autentificación debido a que se seguirán desarrollando nuevas formas de proteger la información y verificar la identidad de los usuarios que solicitan algún servicio.

ANEXO 1

Este dialogo proporciona un relato ficticio del diseño de un sistema de autentificación para redes abiertas llamado "*Charon*". En el transcurso del dialogo, los personajes Athena y Euripides descubren los problemas de seguridad inherentes en un ambiente de red abierto [16], [17], [18].

PERSONAJES

Athena	Desarrolladora de sistemas joven y prometedora
Eurípides	Un desarrollador experimentado y loco

ESCENA I

En un área de cubículos Athena y Eurípides están trabajando en terminales vecinas.

Athena	Oye Rip este sistema de tiempo compartido es un fastidio. No puedo trabajar porque todo mundo esta conectado.
Eurípides	No vengas a quejarte conmigo. Yo solo trabajo aquí.
Athena	¿Sabes qué necesitamos?, necesitamos darle a cada quien su propia estación de trabajo de manera que no se tengan que preocupar de compartir ciclos de proceso. Y usaríamos una red para conectar a todas las estaciones de trabajo para que se puedan comunicar todos.
Eurípides	Bien, a si es que.... ¿Cuantas necesitas?, ¿mas o menos 1000 estaciones de trabajo?
Athena	Más o menos.
Eurípides	¿Conoces el tamaño de un disco duro de una estación de trabajo típica?, no hay espacio suficiente para todo el software que tienes en una maquina de tiempo compartido.
Athena	Si, pero podemos solucionar esto. Podemos guardar copias del software en varias máquinas servidores. Cuando tú te conectes en una estación de trabajo, la estación de trabajo va a accesar este software haciendo una conexión de red con uno de los servidores. De esta manera permites que una gran cantidad de estaciones de trabajo utilice la misma copia del software, y así las actualizaciones

	del software son sencillas. No tienes que andar cambiando software de estación en estación. Solo modificar el software de los servidores.
Eurípides	Está bien. ¿Pero que vas a hacer con los archivos personales? Con los sistemas de tiempo compartido puedes conectarte y accesar a mis archivos desde cualquier terminal que este conectada al sistema. ¿Podré subir a cualquier estación de trabajo y automáticamente accesar a mis archivos? ¿ O tengo que hacerle como un usuario de una PC, guardar mis archivos en un diskette?, Espero que no.
Athena	Pienso que podemos usar otras máquinas para proveer almacenamiento de archivos personales, y así puedes conectarte a cualquier estación de trabajo y accesar a tus archivos
Eurípides	¿Y qué pasa con la impresión?, ¿toda estación de trabajo tiene su propia impresora?, ¿y el correo electrónico?, ¿Como vas a distribuir el correo a todas estas estaciones de trabajo?
Athena:	Ah... Obviamente no tenemos dinero para dar a todos una impresora, pero podríamos tener maquinas dedicadas al servicio de impresión, es decir envías un trabajo a la impresora servidor, y esta lo imprime por ti. Podrías hacer este tipo de cosas con el correo. Tener una maquina dedicada al servicio de correo, si tú quieres tu correo, te conectas al servidor de correo y lo recoges.
Eurípides:	Tu sistema de estación de trabajo realmente suena bien. ¿Cuándo puedo tener el mío, que tengo que hacer? Voy averiguar tu *username*, y tener mi estación de trabajo pensando, entonces que yo soy tú. Después me voy a conectar al servido de correo y voy a recoger tu correo, voy a conectarme al servidor de archivos y borrar tus archivos, y…
Athena	¿Qué puedes hacer que?
Eurípides:	¡Claro!. ¿Cómo van a saber estos servidores de red que yo no soy tu?
Athena:	No se, pero creo que necesito pensar en hacer algo.
Eurípides:	Déjame saberlo cuando lo tengas.

ESCENA II

La mañana siguiente en la oficina de Eurípides. Eurípides esta sentado en su escritorio, leyendo su correo. Athena toca la puerta.

Athena:	Ya se como asegurar un ambiente de red abierto de tal manera que una persona tan aprovechada como tú no pueda usar los servios de red con el nombre de otra persona.
Eurípides	¿Es eso? Toma asiento
Athena:	Antes de describirlo, ¿puedo poner una condición acerca de esta discusión?
Eurípides:	¿Cuál es tu condición?
Athena:	Bien, supón que digo algo como lo siguiente: "Quiero mi correo electrónico, a si que contacto al servidor de correo y le pido que mande el correo a mi estación de trabajo". En realidad yo no soy la entidad que contacta el servidor de correo. Estoy usando un programa que accesa al servidor de correo y recupera mi correo, un programa que es un CLIENTE del programa del servicio de correo. Pero no quiero decir "que el cliente hace esto y aquello" cada vez que me refiero a una transacción entre el usuario y el servidor de red. Yo había dicho "yo hago esto y esto", teniendo en mente desde luego que un programa cliente esta haciendo cosas en mi representación. ¿Te parece bien?
Eurípides:	Seguro. No hay problema.
Athena:	Bueno, esta bien, Voy empezar diciendo el problema que ya he resuelto. En un ambiente de red abierto, las maquinas que proveen servicios debes ser capaces de confirmar la identidad de la gente que requiere servicio, si contacto el servidor de correo y le pido mi correo, el programa de servicio debe ser capaz de identificar que soy quien digo ser, ¿de acuerdo?
Euripides	De acuerdo.
Athena:	Tú podrías resolver el problema tontamente, requiriendo que el servidor de correo pida un *password* antes de poder usarlo. Yo pruebo quien soy al servidor dándole mi *password*.
Eurípides:	Eso esta tontamente bien. En un sistema como ese. Cada servidor debe saber tu *password*. Si la red tiene mil usuarios, cada servidor tiene que saber mil *password*. Si quieres cambiar tu *password* tienes que contactar a todos los servidores y notificarles el cambio. Pensare que tu sistema no es así de estúpido.
Athena:	Mi sistema no es estúpido. Funciona así: No solo la gente tiene *password*, los servicios tienen *password* también, cada usuario sabe su *password*, cada programa de servicio sabe su *password* y existe un **SERVICIO DE AUTENTIFICACIÒN** que sabe **TODOS** los *password*, cada *password* de cada usuario y cada *password* de cada servicio, este servicio de autentificación guarda todos los *password*

	en una sola base de datos centralizada.
Eurípides:	¿Tienes un nombre para este servicio de autentificación?
Athena:	No he pensado en uno todavía, ¿tienes alguna idea?
Eurípides:	¿Cuál es el nombre de aquel tipo quien transporta a los muertos a través del Río Styx?
Athena:	¿*Charon*?
Eurípides:	Si es él. El no te cruzará el río hasta que puedas probar tu identidad.
Athena:	Ahí vas de nuevo Rip, tratando de rescribir la mitología Griega otra vez. A *Charon* no le interesa tu identidad, el sólo quiere asegurarse de que estas muerto.
Eurípides	¿Tienes un mejor nombre?
Athena:	No, no realmente.
Eurípides	Entonces llamemos al servicio de autentificación "*Charon*".
Athena:	OK, creo que solo describiré el sistema, ¿verdad? Digamos que tú quieres usar un servicio, el servicio de correo. En mi sistema tu no puedes usar un servicio a menos que, ¡ah!, *Charon* le diga al servicio que tu eres quien dices que eres. Y tú no puedes recibir el permiso para usar un servicio hasta que te hayas autentificado con *Charon*. Cuando tú requieres a *Charon* que te autentifique, tienes que decirle a *Charon* el servicio para el cual quieres la autorización. Si quieres utilizar el servicio de correo debes decírselo a *Charon*. *Charon* te pide que pruebes tu identidad. Tú lo haces dándole tu *password* secreto. *Charon* toma tu *password* y lo compara con el que tiene registrado en su base de datos. Si los dos *passwords* concuerdan, *Charon* considera que tu identidad ha sido aprobada. *Charon* ahora tiene que convencer al servidor de correo de que tú eres, quien dices que eres. Debido que *Charon* sabe todos los *password* de los servicios, sabe el *password* del servicio de correo. Es concebible de que *Charon* pudiera darte el *password*, el cual puedes mandar el servicio de correo como prueba de que tú te haz autentificado ante *Charon*. El problema es, que *Charon* no puede darte el *password* directamente porque entonces tú lo sabrías. La siguiente vez que tú quisieras correo podrías darle la vuelta a *Charon* y usar el servidor de correo sin identificarte correctamente. Inclusive podrías pretender ser otra persona y usar el servidor de correo con el nombre de otra persona. Así que en lugar de darte el *password* del servidor de correo *Charon* te da un ***ticket*** de servicio de correo. Este ticket contiene una

	versión de tu *username* que ha sido **ENCRIPTADO USANDO EL *PASSWORD* DEL SERVIDOR DE CORREO.** Ahora con tu ticket en mano, puedes pedir al servidor de correo tu correo. Haces tu petición diciéndole al servidor quien eres, y entregándole el ticket que prueba que tú eres quien dices que eres. El servidor usa su *password* para desencriptar el *ticket*, y si el *ticket* se desencripta apropiadamente el servidor va obtener el nombre el *username* que *Charon* puso en el *ticket.* El servido compare este nombre con el nombre que tú enviaste con el *ticket*. Si los nombres coinciden el servidor de correo considera que tu entidad ha sido probada y procede a darte tu correo. ¿Qué piensas de esas manzanas?
Eurípides	Tengo algunas preguntas.
Athena:	Lo imagine, bueno adelante.
Eurípides	Cuando un programa de servido desencripta un *ticket*, ¿Cómo sabe que lo ha encriptado apropiadamente?
Athena:	No lo se.
Eurípides	Quizás deberías incluir el nombre del servicio en el *ticket*, de esa manera cuando un servido desencripte un *ticket* puede medir su éxito, si encuentra su nombre en el nombre en el *ticket* desencriptado.
Athena:	Eso suena bien. Entonces un *ticket* se vería más o menos así.... (Ella escribe sobre un pedazo e papel) ***TICKET - {username: servicename}***
Eurípides	¿Así es que el *ticket* de servicio contiene solo tu nombre de usuario y el nombre del servicio?
Athena:	Encriptado con el *password* de servicio.
Eurípides	No creo que sea suficiente información para hacer el *ticket* seguro.
Athena:	¿Qué quieres decir?
Eurípides	Supongamos que le pides a *Charon* un *ticket* para el servicio de correo. *Charon* prepara ese *ticket* de manera que tiene tu *username* "tina" en el. Supón que yo copio ese *ticket* en el camino a través de la red desde *Charon* hasta ti. Supón que convencía a mi insegura estación de trabajo de que mi nombre de usuario es "tina". El programa Cliente de correo en mi estación de trabajo piensa que soy tú. En tu nombre, el programa envía el *ticket* robado al servidor de correo. El servidor desencripta el *ticket* y ve que es valido el *username* en el *ticket* coincide con el *username* quien envió el *ticket*. El servido de correo me da tu correo...

Athena:	¡Oh!. Eso no esta muy bien.
Eurípides	Pero creo que se una manera para arreglar este problema o por lo menos para proveer un arreglo parcial a el. Creo que *Charon* deberá incluir más información en el *ticket* de servicio que produzca. Además del *username*, el *ticket* deberá incluir la **DIRECCIÓN DE RED** desde la cual es usuario pidió a *Charon* el *ticket*. Esto te da un nivel más información adicional Lo ilustraré. Supón que me robo tu *ticket* de correo en este momento. El *ticket* tiene la dirección de red de tu estación de trabajo, y esta dirección no coincide con la dirección de mi estación de trabajo. En tu nombre yo le mando el *ticket* robado al servidor de correo. El programa servidor extrae el *username* y la dirección de red del *ticket* e intenta comparar esa información con el *username* y la dirección de red de la entidad que envío el *ticket*. El *username* coincide pero la dirección de red no. El servidor rechaza el *ticket* porque obviamente fue robado.
Athena:	Bravo, bravo!!! Quisiera haber ideado eso.
Eurípides	¿Para eso estoy no?
Athena:	Entonces el diseño revisado del *ticket* se ve así: Ella escribe lo siguiente en un pizarrón ***TICKET - {username: ws_address: servicename}*** Ahora estoy realmente emocionada. Construyamos un sistema *Charon* y veamos si funciona!!
Eurípides	No tan rápido. Tengo algunas otras preguntas acerca de tu sistema.
Athena:	Bueno, dispara.
Eurípides	Suena como que tengo que pedir un *ticket* cada vez que quiera usar un servicio. Si tengo un día de trabajo completo, probablemente necesite checar mi correo más de una vez. ¿Tengo que sacar un *ticket* nuevo cada vez que quiera mi correo?. Si eso es cierto, no me gusta tu sistema.
Athena:	¡Ah!... bueno, no veo porque los *tickets* no puedan ser reutilizables. Si el servidor de correo te da un *ticket*, debe ser capaz de usarlo uno y otra vez. Por ejemplo, cuando el programa cliente de correo hace una petición de servicio en tu nombre, le manda una COPIA del *ticket* al servidor de correo.
Eurípides	Eso esta mejor. Pero todavía hay problemas capaces de implicar que tengo que darle a *Charon* mi *password* cada vez que quiero usar un servicio para el cual no tengo un *ticket*. Me conecto al sistema y quiero accesar a mis archivos. Le mando una requisición a *Charon* para el *ticket* apropiado y esto significa que tengo que usar mi

	password. Entonces quiero leer mi correo. Otro requerimiento a *Charon*, y tengo que darle mi *password* otra vez. Ahora supón que quiero enviar mis mensajes de correo al servidor de impresión, otra requisición a *charon* y, bueno ya checaste la escena.
Athena:	Si, entiendo.
Eurípides:	Y por si eso fuera poco, considera esto: suena como cuando tú te autentificas ante *Charon*, tú mandas tu *password* sobre la red en texto. Gente inteligente como tu verdaderamente podría monitorear la red y robar copias de los *password* de la gente. Si tengo tu *password*, puedo usar cualquier servicio en tu nombre. *(*Athea suspira)
Athena:	Estos son problemas serios. Creo que necesito regresar a mi escritorio

ESCENA III

La mañana siguiente. Athena encuentra a Eurípides en el área del café. Ella le da una palmadita en el hombro mientras el llena su taza.

Athena:	Tengo una nueva versión de Charon que resuelve nuestros problemas.
Eurípides	¿De verdad? Eso estuvo rápido.
Athena:	Bien, tú sabes, que los problemas de esta naturaleza me mantienen despierta toda la noche.
Eurípides	Debe ser tu conciencia culpable. ¿Nos vamos a ese pequeño cuarto de conferencias? .
Athena:	¿Por qué no? (Los dos se dirigieron al pequeño cuarto de conferencias)
Eurípides	Empezaré describiendo los problemas nuevamente. Pero los invertiré de tal manera que serán los requerimientos del sistema Athena aclara su garganta.
Athena:	Primer requerimiento: Los usuarios solo tienen que ingresar sus *passwords* una sola vez, al comienzo de sus sesiones en su estación de trabajo. Este requerimiento implica no meter tu password cada vez que necesites un nuevo ticket de servicio. Segundo requerimiento: Los passwords no deberán ser enviados sobre la red en texto

Eurípides	OK
Athena:	Empezaré con el primer requerimiento: Deberás solo usar tu *password* una sola vez. He cumplido este requerimiento inventando un nuevo servicio de red. Es llamado el servicio de "**entrega de ticket**", un servicio que emite *tickets* de *Charon* a usuarios quienes hayan probado su identidad a *Charon*, tú puedes usar este servicio de entrega de *tickets* si tienes un *ticket* para esto, es decir , un *ticket* para emisión de *tickets*. El servicio de entrega de tickets en realidad es una versión de *Charon* tanto que tiene acceso a la base de datos de *Charon* es una parte de *Charon* que permite autentificarte con un *ticket* en lugar de un *password*. De todas maneras, el sistema de autentificación ahora trabaja de manera siguiente: tú te conectas en una estación de trabajo y usas un programa llamado *Kinit* para contactar al servidor de *Chanon*. Tu pruebas tu identidad a Charon, y el programa *Kini*t te da un *ticket* para entrega de *tickets*. Ahora digamos que quieres obtener tu correo del servidor de correo. No tienes un *ticket* para servicio de correo, todavía, así que usas el *ticket* de entrega de *tickets* para que obtenga el *ticket* del servidor de correo por ti. No tienes que usar tu *password* para obtener el nuevo *ticket.*
Eurípides:	¿Tengo que tener un nuevo *ticket* de entrega de *tickets* cada vez que necesite usar otro servicio red?
Athena:	No recuerda que acordamos que los *tickets* puedes ser reutilizados. Una vez que hayas adquirido un *ticket* para entrega de tickets no necesitas obtener otro. Tú usas el *ticket* para entrega de *tickets* para obtener los otros *tickets* que necesites.
Eurípides:	OK, eso tiene sentido. Y ya que puedes reutilizar los *tickets*, una vez que el servicio de entrega de *tickets* te ha dado un *ticket*, para un servicio en particular no necesitas obtener ese *ticket* en particular otra vez.
Athena:	Si, ¿No es elegante?
Eurípides:	OK, lo compro hasta ahora... Siempre y cuando no tengas que enviar tu password en texto sobre la red cuando obtengas el *ticket* para entrega de *tickets*.
Athena:	Como dije, he resuelto ese problema también. La cosa es, cuando digo que tienes que contactar a Charon para obtener el *ticket* para entrega de *tickets*, sonó como si tuvieras que enviar tu password en texto sobre la red, al servidor de Charon. Pero no tiene que ser de esa manera. Esto es lo que realmente pasa. Cuando tú uses el programa *Kinit*

	para obtener el *ticket* para entrega de *tickets*, *Kinit* no manda tu *password* al servidor de *Charon*. *Knit* manda solo tu *username.*
Eurípides:	Bien.
Athena:	*Charon* utiliza el *username* para buscar tu *password.* En seguida *Charon* construye un paquete de datos que contiene el *ticket* para entrega de *tickets*. Antes de que mande el paquete, *Charon* utiliza tu *password* para encriptar el contenido del paquete. Tu estación de trabajo recibe el paquete del *ticket*, tu ingresas tu *password*, *Kinit* intenta desencriptar el *ticket* con el *password* que ingresaste, Si *Kinit* tiene éxito, te haz autentificado ante *Charon*. Ahora posees un *ticket* para entrega de *tickets*, y ese *ticket* puede darte los demás *tickets* que requieras. ¿Qué tal esas modernas ideas?
Eurípides:	No se... Estoy tratado de pensar. Tú sabes, pienso que las partes del sistema que me acabas de describir funcionan muy bien. Tu sistema requiere que me autentifique solo una vez. Después de eso *Charon* me puede emitir *tickets* de servicio sin que yo me de cuenta, muy bien, muy bien en ese sentido, Pero hay algo acerca del diseño del *ticket* de servicio que me incomoda de alguna manera. Tiene que ver con el hecho de que los *tickets* son reutilizables, Ahora estoy de acuerdo que tienen que ser reutilizables, pero los *tickets* reutilizables son por su naturaleza muy peligrosos.
Athena:	¿Qué quieres decir?.
Eurípides:	Míralo de esta forma. Supón que esta usando una estación de trabajo insegura. En el transcurso de tu sesión, tú adquieres un *ticket* de servicio de correo, un *ticket* de servicio de impresión, y un *ticket* de servicio de archivos. Supón que inadvertidamente dejas esos *tickets* e la estación de trabajo cuando te sales del sistema. Ahora supón que me conecto en la estación de trabajo y encuentro esos *tickets.* Y me siento con ganas de causar problemas, entonces hago que la estación de trabajo piense que soy tu. Debido que los *tickets* están hechos a tu nombre, puedo usar el programa cliente de correo para accesar a tu correo, puedo usar el servicio de archivos y borrar tus archivos, y puedo usar el comando de impresión para causar cargos a tu cuenta. Y todo gracias a esos *tickets* que se dejaron accidentalmente. Y nada puede detenerme de copias estos *tickets* a mi lugar. Puedo seguir usándolos por toda la eternidad.
Athena:	Pero es fácil de arreglar. Solo escribimos un programa que destruya los *tickets* de un usuario cada vez que el usuario se desconecte. No puedes usar *tickets* que se han destruido.
Eurípides:	Bueno obviamente tu sistema debe tener un programa para destruir

	los *tickets*, pero es tonto confiar en los usuarios para que hagan una cosa así. No puedes contar con que los usuarios recuerden destruir sus *tickets*, cada vez que terminen una sesión. Y aunque pudieras confiar en los usuarios para destruir sus *tickets*, considera el siguiente escenario. Tengo un programa que vigila la red y copia los *tickets* de servicio cuando están viajando en la red. Supón que quiero hacerte mi víctima. Espero a que inicies una sesión, activo mi programa y copio un montón de tus *tickets.* Espero que termines tu sesión, y eventualmente tú te sales y te vas. Yo juego con el *software* de red de mi estación de trabajo y cambio su dirección de tal manera que concuerde con la dirección de la estación de trabajo que estabas usando cuando te robe los *tickets*. Hago creer a mi estación de trabajo que yo soy tu. Tengo tus *tickets*, tu username, y la dirección de red correcta. Puedo mandar estos *tickets* y usar servicios en tu nombre. No importa que hayas destruido tus *tickets* antes de haber terminado tu sesión. Los *tickets* que he robado son validos tanto como tenga cuidado para usarlos, ya que tu diseño actual de *tickets* no pone un límite en el número de veces que puedes reutilizar un *ticket*, o cuanto tiempo permanece vigente un *ticket*.
Athena:	Oh veo lo que estas diciendo!, los *tickets* no pueden ser validos por siempre, porque constituirían un gran riesgo en la seguridad. Debemos restringir la longitud de tiempo por el cual un *ticket* puede ser usado, quizás darle a cada *ticket* alguna clase de fecha de expiración.
Eurípides:	Exactamente, pienso que cada *tickets* necesita tener dos piezas adicionales de información, una duración media que indique la longitud de tiempo para la cual el *ticket* es válido, un sello de tiempo que indique la fecha y la hora en la cual Charon emitió el *ticket*. Así que el ticket se vera algo así : Eurípides se dirige al pizarrón escribe lo siguiente: ***TICKET {username: address: servicename: lifespan: timestamp}*** Ahora cuando un servicio desencripte *tickets*, compara el username y la dirección del *ticket* contra el username y la dirección de la persona que esta enviando el *ticket*, y usa la información de la duración media y el sello de tiempo para ver si el *ticket* ha expirado.
Athena:	Bien, ¿Qué clase de tiempo de vida debería tener un *ticket* de servicio típico?.
Eurípides:	No lo se. Probablemente la longitud de una sesión típica en una estación de trabajo. Digamos, ocho horas.

Athena:	Así que si me siento en mi estación de trabajo por más de ocho horas, todos mis *tickets* expiran, eso incluye el *ticket* para entrega de *tickets*. Así que tengo reutentificarme después de ocho horas.
Eurípides:	Eso no es irrazonable, ¿o si?.
Athena:	Pues creo que no, entonces ya quedamos, los *tickets* expiran después de ocho horas. Ahora te tengo una pregunta, imagínate que copie tus *tickets* de la red...
Eurípides:	¡Ahh... Tina!. ¿Tú no harías eso verdad?.
Athena:	Esto es solo por el bien del argumento. Ya copie tus *tickets*. Ahora espero a que te vayas. Imagínate, que tienes una cita con el doctor, o una clase, o algo por el estilo, y terminas tu sesión después de un par de horas. Tú eres un chico listo y haz destruido tus copias de los *tickets* antes de irte. Pero te robe los *tickets* y son buenos por cerca de seis horas. Eso me da un buen de tiempo para jugar con tus archivos e imprimir mil copias de quien sabe que, a tu nombre. ¿Checas?, el arreglo del sello de tiempo y del periodo de expiración funcionan bien en el evento de que un ladrón de *tickets* elija mandar un *ticket* después de que ha expirado. Si el ladrón puede mandar el *ticket* antes de eso...
Eurípides:	Mmm, bueno... Estas en lo correcto.
Athena:	Creo que nos hemos metido en un problema mayor, ahh... (Pausa)
Eurípides:	Creo que eso significa que vas a estar ocupada esta noche. ¿Más café?.
Athena:	¿Por qué no?.

ESCENA IV

A la mañana siguiente, en la oficina de Eurípides. Athena toca la puerta.

Eurípides:	Esta mañana tienes ojeras.
Athena:	Bueno, tú sabes, fue una de esas largas noches.
Eurípides:	¿Resolviste el problema?.
Athena:	Eso creo.
Eurípides:	Toma asiento.

Athena:	Como ya es costumbre, me sentí forzada a volver a plantear el problema. Los *tickets* son reutilizables dentro de un periodo de tiempo limitado, digamos, ocho horas. Si alguien roba tus *tickets* y elige mandarlos antes de que expire no podemos hacer nada para detenerlo.
Eurípides:	Ese es el problema.
Athena:	Podríamos solucionar el problema si diseñáramos los *tickets* de manera que no fueran reutilizables.
Eurípides:	Pero entonces tú tendrías que tener un *ticket* nuevo cada vez que quisieras usar un servicio de la red.
Athena:	Cierto, esa es a lo más una solución tonta. (*pausa*) Ah, ¿y como procedo con mi argumento? (*ella piensa por un momento*). Bien, voy a replantear el problema otra vez, esta vez en la forma de un requerimiento. Un servicio de red debe ser capaz de probar que la persona que esta usando un *ticket* es la misma para la cual el *ticket* fue expedido. Déjame trazar de nuevo el proceso de autentificación y ver si puedo encontrar una manera apropiada para ilustrar mi solución a este problema. Quiero usar un cierto servicio de red. Acceso ese servicio iniciando un programa cliente en mi estación de trabajo. El cliente manda tres cosas al servidor, mi nombre, la dirección de red de mi estación de trabajo, y el *ticket* de servicio apropiado. El ticket contiene el nombre de la persona a quien fue emitido y la dirección de la estación de trabajo que la persona estaba usando cuando el o ella adquirió el *ticket*. Además contiene una fecha de expiración en forma de un tiempo de vida y un sello de tiempo. Toda esta información ha sido encriptada con el *password* del servicio *Charon.* Nuestro esquema de autentificación actual descansa sobre las siguientes pruebas: • ¿El servicio puede desencriptar el *ticket*? • ¿Ha expirado el *ticket*? • ¿Coinciden el *username* y la dirección de la estación de trabajo especificadas en el *ticket* con el nombre y dirección de la persona que lo envió? ¿Qué arrojan estas pruebas?. **La primera prueba**, dice si el *ticket* vino o no vino de *Charon*. Si el *ticket* no puede ser desencriptado no viene de *Charon*. El *Charon* real habría encriptado el *ticket* con el *password* del servicio. *Charon* y el servicio son las únicas dos

	entidades que saben el *password* del servicio. Si el *ticket* se desencripta correctamente, el servicio sabe si vino de *Charon* realmente. Estas pruebas previenen que alguien construya tickets falsos de *Charon*. **La segunda prueba** checa el tiempo de vida y el sello de tiempo. Si ya expiro el servicio rechaza el *ticket*. Esta prueba detiene a gente usando *tickets* viejos, *tickets* que tal vez fueron robados. La tercera prueba checa el username y la dirección del *ticket* contra el *username* y la dirección de la persona especificada en el *ticket*. Si la prueba falla el usuario del *ticket* ha obtenido el *ticket* de otra persona. El *ticket* es rechazado. Si los nombre y las direcciones coinciden, ¿qué se probo?. Nada, los chicos malos pueden robar *tickets* de la red, cambiar la dirección de su estación de trabajo y su *username* y saquear los recursos de otras personas. Como señale ayer, los *tickets* pueden ser reenviados siempre y cuando no hayan expirado. Pueden ser reenviados porque un servicio no puede determinar que la persona que esta enviando el ticket es su dueño legítimo. El servicio no puede hacer esta determinación debido a que no comparte un secreto con el usuario. Veámoslo de esta forma, si estoy vigilando Elsinore, tú sabes, el castillo en Hamlet, y se supone que me vas a relevar, no te voy a dejar tomar mi lugar hasta que me des el *password* correcto. Ese es el caso donde los dos compartimos un secreto. Y es probablemente un secreto que alguien mas invento para cualquier vigía. Así que estaba pensando anoche, ¿porque no hacer que *Charon* invente un *password* para el dueño legitimo del ticket para compartirlo con el servicio?, *Charon* le da una copia de esta **llave de sesión** al servicio, y una copia al usuario. Cuando el servicio recibe un *ticket* de un usuario. Puede usar la llave de sesión para probar la identidad del usuario.
Eurípides:	Espera un segundo. ¿Como les va a dar *Charon* la llave de sesión a ambas partes?.
Athena:	El dueño del *ticket* obtiene la llave de sesión como parte de la respuesta de *Charon*. Como esto: ***CHARON REPLY - [sessionkey | ticket]*** La copia de la llave de sesión del servicio viene dentro del ticket, y el servio obtiene la llave cuando desencripta el ticket así es que el ticket se ve así: ***TICKET - {sessionkey:username:address:servicename:lifespan:timestamp}***

	Cuando quieras obtener un servicio, el programa Cliente que tú inicias construye a lo que yo le llamo **AUTENTIFICADOR**. El autentificador contiene tu nombre y tu dirección. El cliente encripta esta información con la llave de sesión, la copia de la llave de sesión que tu recibiste cuando pediste ese ticket de la siguiente manera: ***AUTHENTICATOR - {username:address} encrypted with session key*** Después de construir el autentificador, el cliente lo manda con el *ticket* al servicio. El servicio no puede desencriptar el autentificador todavía porque no tiene la llave de sesión. Esa llave esta en el *ticket* así que el servicio primero desencripta el *ticket*. Después de que se descripta el *ticket*, el servicio termina con la siguiente información: • El periodo de vida del *ticket* y el sello de tiempo. • El nombre del dueño del *ticket*. • La dirección de red del dueño del *ticket* . • La llave de sesión. El servicio checa si el *ticket* ha expirado. Si todo va bien en ese sentido, El servicio entonces utiliza la llave de sesión para desencriptar el autentificador. Si la desencriptación procede sin obstáculo el servicio termina con un *username* y una dirección de red. El servicio prueba esta información contra el nombre y dirección encontrados en el *ticket*, Y el nombre y dirección de la persona quien envió el *ticket* y el autentificador. Si todo concuerda, el servicio ha determinado que quien envió el *ticket* es el verdadero dueño del *ticket.* (Athena se detiene, aclara su garganta y toma café) Creo que la llave de sesión y el autentificador se hacen cargo del problema del reenvío.
Eurípides:	Quizás pero me pregunto... ¿Para violar esta versión del sistema, debe tener el autentificador apropiado para el servicio?
Athena:	No. Debes de tener el autentificador y el *ticket* para el servicio. El autentificador no tiene valor sin el *ticket* porque el servicio no puede desencriptar el autentificador sin tener primero la llave de sesión apropiada, y el servicio no puede obtener la llave de sesión apropiada sin primero desencriptar el *ticket.*
Eurípides:	OK. Entiendo, pero no dijiste que cuando el programa cliente contacta al servidor,¿ le manda el *ticket* y el autentificador juntos?
Athena:	Si creo que eso fue lo que dije.
Eurípides:	Si eso es lo que pasa realmente, ¿que me impide robar el ticket y el autentificador al mismo tiempo?. Estoy seguro que podría escribir un programa para hacer el trabajo. Si tengo el *ticket* y su autentificador, creo que puede usar los dos siempre y cuando el *ticket* no expire.

	Solo tengo que cambiar mi dirección y mi *username* apropiadamente, ¿cierto?.
Athena:	Cierto, que desesperante.
Eurípides:	¡Espera, espera, espera! No es tanto problema. Los *tickets* son reutilizables siempre y cuando no hayan expirado, pero eso no significa que los autentificadores tienen que ser reutilizables. Supón que diseñamos el sistema de manera que los autentificadores puedan ser usados sólo una vez.¿Eso nos da algo?.
Athena:	Bueno, podría. Veamos el programa cliente construye el autentificador, entonces lo manda con el *ticket* al servicio. Tu copias el *ticket* y el autentificador mientras se mueven de mis estación de trabajo al servidor. Pero el *ticket* y el autentificador llegan al servidor antes de que tú puedas enviar tus copias. Si el autentificador puede usarse sólo una vez tu copia no es buena y tú pierdes cuando tratas de reenviar tu *ticket* y tu autentificador. Eso es un alivio. Así que todo lo que tenemos que hacer es inventar una manera en que el autentificador sea útil una sola vez.
Eurípides:	No hay problema. Pongamos un tiempo de vida y un sello de tiempo en el. Supón que cada autentificador tiene un tiempo de vida de un par de minutos. Cuando quieres usar un servicio tu programa cliente construye el autentificador, lo sella con la hora actual y se envía junto con el *ticket* al servidor. El servidor recibe el *ticket* y el autentificador y se pone a trabajar. Cuando el servidor desencripta el autentificador checa su tiempo de vida y el sello de la hora. Si el autentificador no ha expirado y todo lo demás checa, el servidor te considera autentificado. Imagina que copie el autentificador y el *ticket* mientras cruzaban la red. Tengo que cambiar mi dirección de red y mi *username*, y lo tengo que hacer en un par de minutos. Esta muy difícil que se pueda. De hecho no creo que sea posible. A menos... Bueno aquí hay un problema potencial. Supón que en lugar de copiar el *ticket* y el autentificador mientras viaja de tu estación de trabajo al servidor copia el paquete original del *ticket* que viene de *Charon*, el paquete que recibes cuando le pides a *Charon* un *ticket*. Este paquete según recuerdo, tiene dos copias de la llave de sesión, una para ti y otra para el servicio. La del servicio esta escondida en el *ticket* y no la puedes obtener, pero que hay de la otra, la que usas para construir autentificadores. Si puedo obtener esa copia de la llave de sesión, entonces puedo construir mis propios autentificadores, si los puedo construir, puedo irrumpir en el sistema
Athena:	Eso es algo que pensé anoche, pero entonces seguí el proceso de adquisición de *tickets* y encontré que no es posible robar

	autentificadores de esa manera. Tú te sientas en una estación de trabajo y utilizas el programa *Kinit* para obtener tu *ticket* para entrega de tickets. *Kinit* te pide tu username, y después de que lo ingresas, *Kinit* le manda el nombre a *Charon*. *Charon* usa tu nombre para buscar tu *password*, entonces procede a construir un *ticket* de entrega de *ticket* para ti. Como parte de este proceso, *Charon* crea una llave de sesión que tú compartirás con el servicio de entrega de *ticket*. *Charon* pone una copia de la llave de sesión en el *ticket* de entrega de *ticket*, y pone tu copia en el paquete del *ticket* que estas a punto de recibir. Pero antes de que te envíe este paquete, *Charon* encripta todo con tu *password*. *Charon* envía el paquete a través de la red. Alguien puede copiar el paquete en su camino, pero no pueden hacer nada con el, porque ha sido encriptado con tu *password*, específicamente, nadie puede robar la llave de sesión de la entrega de *ticket*. *Kinit* recibe el paquete del *ticket* y te pide un *password*, el cual tú ingresas. Si tú ingresas el *password* correcto *Kinit* puede desencriptar el paquete y darte tu copia de la llave de sesión. Ahora que haz hecho cargo del rollo de *Kinit*, quieres obtener tu correo. Tú inicias el programa cliente de correo. Este programa busca un *ticket* de servicio de correo y no lo encuentra. El cliente debe usar el *ticket* para entrega de *ticket* para pedirle al servicio de entrega de *tickets* un *ticket* de servicio de correo. El cliente construye un autentificador para la transacción de entrega de *tickets* y encripta el autentificador con tu copia de la llave de sesión de la entrega de *ticket*. El cliente manda a *Charon* el autentificador, el *ticket* para entrega de *tickets*, tu nombre, la dirección de tu estación de trabajo y el nombre de servicio de correo. El servicio de entrega de correo recibe estas cosas y ejecuta los chequeos de autentificación. Si todo checa apropiadamente. El servicio de entrega de *ticket* finaliza con una copia de la llave de sesión que comparte contigo. Ahora el servicio de entrega de *ticket* te construye un *ticket* de servicio de correo, y durante este proceso, crea una nueva llave de sesión para ti, para compartir con el servicio de correo. El servicio de entrega de *ticket* ahora prepara un paquete de *ticket* para mandarlo a tu estación de trabajo. El paquete contiene el *ticket* y tu copia de la llave de sesión del servicio de correo. Pero antes de que mande el paquete, el servicio de entrega de *ticket* encripta el paquete con su copia de la llave de sesión de la entrega de *ticket*. Hecho esto, el paquete es enviado. Entonces viene el paquete del *ticket* del servicio de correo, a través de la red. Supón que un ogro de la red lo copia en su camino. El ogro no tiene suerte porque el paquete esta encriptado con la llave de sesión de entrega de *ticket*, tú y el servicio de entrega de *ticket* son

	las únicas entidades que conocen esta llave. Ya que el ogro no puede desencriptar el paquete del *ticket* de correo, el ogro no puede descubrir la llave de sesión de correo. Sin esta llave de sesión el ogro no puede usar ninguno de los *ticket* de servicio de correo que e puedan mandar subsecuentemente a través de la red. Creo que estamos a Salvo. ¿Que piensas?.
Eurípides:	Tal vez.
Athena:	Tal vez!. Es todo lo que vas a decir.
Eurípides:	No te enojes. Ya deberías conocerme. Creo que eso significa para mi y para ti que vas a estar despierta a mitad de la noche.
Athena:	¡Pthhhhh!
Eurípides:	Bien, tres cuartos de noche. Hasta el momento esta empezando a sonar aceptable. Esta llave de sesión resuelve un problema que pensé anoche: El problema de la autentificación mutua. ¿Te importa si hablo por un minuto?
Athena:	No hay problema.
Eurípides:	Muy amable. Anoche, mientras danzaban en tu cabeza visiones de llave de sesión y autentificadores. Estaba tratando de encontrar nuevos problemas en el sistema, y encontré uno que es bastante serio. Te lo ilustrare en el siguiente escenario. Supón que estas harto de tu trabajo actual y haz decido cambiarte, quieres imprimir tu *curriculum* en la impresora láser de tu compañía de manera que los buscadores de talentos y empleadores potenciales puedan tomar nota de la clase que tienes. Así es que ingresas al comando de impresión, y lo direccionas al servidor de impresión adecuado. El comando obtiene el *ticket* de servicio apropiado, si no lo tiene todavía, entonces manda el *ticket* en tu nombre al servidor de impresión. Al menos eso es lo que tu crees que esta pensado. Tú no sabes que el requerimiento es direccionado al servidor de impresión correcto. Imagina que algún *hacker* sin escrúpulos (digamos que es tu jefe) ha arreglado el sistema de manera tal que redirecciona tu petición y el *ticket* al servidor de impresión a su oficina. El programa de servicio de impresión no le importa el *ticket* o su contenido. Tira el *ticket* y envía un mensaje a tu estación de trabajo indicando que el *ticket* pasa, y que el servidor esta listo, esperando imprimir tu trabajo. El comando de impresión manda el trabajo al servidor de impresión fraudulento y el enemigo termina con tu *curriculum*. Mostraré el problema por medio del contraste. Sin llaves de sesión y autentificadores, *Charon* pueden proteger a sus servidores de usuarios falsos, pero no puede proteger a loa usuarios de servidores falsos. El sistema necesita una manera para que los programas

	cliente autentifique el servidor antes de enviar información al servicio. El sistema debe permitir autentificación mutua Pero la llave de sesión resuelve este problema siempre y cuando diseñes los programas cliente apropiadamente. Regresando al escenario del servidor de impresión, Quiero un programa Cliente de impresión que se asegure de mandar los trabajos al servicio legítimo. Esto es lo que hace un programa así. Ingreso el programa de impresión y le doy el nombre del archivo, el nombre de mi *curriculum*. Asumo que tengo un *ticket* de servicio de impresión y una llave de sesión. El programa cliente usa la llave de sesión para construir un autentificador, Después manda el autentificador y el *ticket* al "supuesto" servidor de impresión. El cliente **NO MANDA** el *curriculum* todavía, espera una respuesta del servicio. El servicio real recibe el ticket y el autentificador, desencripta el *ticket* y extrae la llave de sesión, entonces usa la llave la sesión para desencriptar el autentificador. Hecho esto, el servicio corre todas las pruebas de autentificación apropiadas. Asumo que las pruebas confirmaron mi identidad. Ahora el servidor prepara un paquete de respuesta de manera que pueda probar su identidad, al programa cliente. Usa su copia de llave de sesión para encriptar el paquete de respuesta, y envía el paquete al cliente que esta esperando. El cliente recibe el paquete e intenta desencriptarlo con mi copia de la llave de sesión. Si el paquete se desencripta apropiadamente y produce el mensaje de respuesta correcto del servidor, mi programa Cliente sabe que el servidor que encriptó el paquete es el servidor real. Ahora el cliente manda el *curriculum* al servicio de impresión. Supón que mi jefe arreglo el sistema de manera tal que su servidor de impresión parece al que yo quiero. Mi Cliente manda el autentificador y el ticket al servicio de impresión y espera su respuesta. El servicio de impresión falso no puede generar la respuesta correcta porque no puede desencriptar el *ticket* y obtener la llave de sesión. Mi cliente no mandará el trabajo hasta que reciba la respuesta correcta. Eventualmente el Cliente se da por vencido. Mi trabajo de impresión no se completa, pero al menos mi *curriculum* no termino en el escritorio del enemigo. Sabes, creo que tenemos bases sólidas sobre las cuales implementar el Sistema de Autentificación *Charon*.
Athena:	Quizá, de cualquier manera, no me gusta el nombre "*Charon*".
Eurípides:	¿No? ¿Desde cuando?
Athena:	Nunca me gusto, por que el nombre no tiene sentido. El otro día estaba hablando de esto con mi tío Hades, y me sugirió otro nombre, el nombre de su perro guardián de tres cabezas

Eurípides:	Oh, te refieres a "***Cerberus***".
Athena:	Comete tu lengua Rip! "***Cerberus***"
Eurípides:	¿No es ese el nombre?
Athena:	Si, ¡si tú fueras romano! Soy una diosa Griega, el es un perro guardián Griego, y su nombre es "**Kerberos**", "**Kerberos**" con K
Eurípides:	Esta bien, esta bien, No dispares rayos, me gusta el nombre. De hecho le queda bien: Adiós *Charon* y hola **Kerberos**.

GLOSARIO

Amenaza	Situación, circunstancia o evento con la capacidad de violar la seguridad o causar daño a los recursos del sistema.
AS	*Authentication Server* – Servidor de Autentificación.
Ataque de réplica	Ataque sobre un sistema de autentificación que consiste en la grabación o copiado de los mensajes válidos enviados anteriormente o de parte de los mismos. Cualquier información de autentificación que sea constante, como la contraseña, la aplicación de una función *hash* unidireccional a una contraseña o incluso datos biométricos transmitidos de forma electrónica; se puede grabar y ser utilizada posteriormente para falsificar mensajes y darles la apariencia de auténticos.
Autentificación	Proceso en el cual se verifica la identidad de un usuario o un proceso.
Autentificador	Registro de dato que contiene información, la cual se demuestra que ha sido recientemente generada utilizando la clave de sesión conocida solamente por el cliente y el servidor solicitado.
Autorización	Es el proceso que determina si se permite o no a una identidad autenticada tener acceso a un recurso solicitado o ejecutar una operación solicitada.
CBC	*Cipher Block Chaining* – Encadenamiento de Bloques Cifrados.
Cifrado	Proceso de transformación criptográfica de datos para producir un texto encriptado de manera que los datos aparezcan de una manera ininteligible.
Cifrado Digráfico	Cifrado en bloques de 2 caracteres.
cipher text	El texto cifrado es la salida de una función de cifrado. El cifrado transforma el texto normal en texto cifrado.
Clave	Secuencia de símbolos que controlan las operaciones de cifrado y descifrado. Los algoritmos de cifrado simétricos utilizan la misma clave para cifrar y descifrar,

	mientras que los algoritmos asimétricos utilizan un par de claves: pública y privada.
Clave de sesión	Clave temporal compartida entre 2 a más principales con un tiempo de vida limitado.
Clave entre reinos	Clave secreta compartida por dos centros de distribución de claves (KDC) en reinos diferentes.
Clave privada	Clave criptográfica utilizada en algoritmos de clave pública, se utilizan para cifrar una clave de sesión simétrica, para firmar digitalmente un mensaje o para descifrar un mensaje que haya sido cifrado con la clave pública correspondiente.
Clave pública	Una clave pública es la mitad pública de un par de claves: pública y privada. Suele utilizarse al descifrar una clave de sesión o una firma digital. La clave pública también se puede utilizar para cifrar un mensaje, garantizando de este modo que sólo pueda descifrar el mensaje la persona con la correspondiente clave privada.
Clave secreta	Clave utilizada en un criptosistema simétrico, que es compartida entre las entidades involucradas en la comunicación.
Cliente	Proceso que solicita y eventualmente obtiene un servicio de red. Los clientes actúan generalmente en nombre de los usuarios.
Confidencialidad	Propiedad de la información no sea revelada o puesta a disposición de partes no autorizadas.
Contabilidad	Propiedad que asegura que las acciones de un principal particular pueden ser seguidas sólo para ese principal.
Control de acceso	Proceso cuyo objetivo es evitar el uso no autorizado de recursos, incluyendo la prevención del uso de recursos de forma no autorizada.
Cracker	Intruso. Individuo con amplios conocimientos en informática que intenta acceder a sistemas computarizados sin autorización. Estos individuos tienen a menudo malas intenciones ya que roban,

	alteran o destruyen la información de un sistema.
Credenciales	Registros de datos necesarios para establecer la supuesta identidad de un principal. En le modelo de Kerberos, las credenciales están formadas por un *ticket* junto con la clave de sesión secreta necesaria para utilizar con éxito, el *ticket* en le proceso de autentificación.
Criptoanálisis	Es la ciencia que trata del encriptamiento de mensajes y para determinar la fuerza de las técnicas de encriptación.
Criptografía	Ciencia de la codificación de los mensajes, de forma que no puedan ser leídos por alguien que no sea el receptor.
Demótica	Sistema de escritura cursiva egipcia derivada de la hierática y utilizada durante la Baja Época para escribir todo tipo de documentos administrativos, familiares y literarios de carácter popular, razón por la cual los griegos le dieron este nombre. La escritura demótica refleja la lengua hablada por los egipcios durante el primer milenio, conocida como lengua demótica.
DES	Criptosistema con clave secreta (*Data Encryption Standard* - Estándar de Cifrado de Datos).
Descifrado	Proceso inverso del cifrado, donde el receptor desencripta los datos cifrados enviados por el emisor.
Estándar	Acuerdo documentado que contiene especificaciones técnicas u otros criterios preciso para ser utilizados, que consiste en regla, guías o definiciones de características para asegurar que los materiales, productos, procesos y servicios adecuados para su propósito.
Firewall	Se trata de un mecanismo de seguridad en Internet frente a accesos no autorizados. Básicamente consiste en un filtro que comprueba la identidad de los paquetes y rechaza todos aquellos que no estén autorizados o correctamente identificados.
Hackers	Experto en informática capaz de entrar en sistemas cuyo acceso es restringido, no necesariamente con

	malas intenciones. No obstante, el término se asocia con el de delincuente informático, e incluye a los cibernautas que realizan operaciones delictivas a través de las redes de computadoras existentes.
Host	Representa una computadora que es servidor en una LAN, con aplicaciones y recursos que pueden ser compartidos por los distintos usuarios pertenecientes a ella.
IDEA	Criptosistema con clave secreta (*International Data Encryption Algorithm* – Algoritmo de Cifrado de Datos Internacional).
Información	Conocimiento comunicado o recibido concerniente a un hecho particular o circunstancia en general, y datos que puedan ser codificados para su procesamiento por una PC o dispositivo en general.
Integridad	Propiedad en la que se asegura que los datos se transfieren desde una fuente a un destino sin alteraciones.
KDC	*Key Distribution Center* - Centro de distribución de claves.
Kerberos	Sistema de autentificación y distribución de llaves desarrollado por el MIT para proteger los intercambios en la red de una universidad o una organización.
LAN	*Local Area Network* – Red de Área Local. Red de datos de alta velocidad y bajo nivel de error que cubre un área geográfica relativamente pequeña (hasta unos pocos miles de metros).
MIT	Massachusetts Institute of Technology – Instituto de Tecnología de Massachusetts.
Modelo cliente-servidor	Término utilizado para describir los sistemas de red de informática distribuida en los cuales las responsabilidades de la transacción se dividen en dos partes: cliente (frontal) y servidor (nodo). Ambos términos (cliente y servidor) pueden aplicarse a programas de software o a dispositivos reales de computación. También llamado informática distribuida.

NetSP	*Network Security Program* – Programa de Seguridad en Redes.
NIST	*National Institute for Standards and Technology* – Instituto Nacional de Estándares y Tecnología.
NSA	*National Security Agency* - Agencia Nacional de Seguridad.
Password	Contraseña. Conjunto de caracteres que permite acceder a un determinado contenido en la red o que sirve para discriminar el acceso de los usuarios.
PCBC	*Plain and Cipher Block Chaining* - Encadenamiento de Bloques Cifrados y Planos.
PGP	*Pretty Good Privacy* – Privacidad muy buena.
plain text	El texto plano es la entrada a una función de cifrado o salida de una función de descifrado.
Pre autentificación	Autentificación previa al intercambio real de la autentificación.
Principal	Entidad humana o sistema registrado y autentificable para una red de computadoras o sistemas distribuidos.
Proceso	Ejecución de un programa que se ejecuta en un sistema determinado.
Protocolo	Especificación del formato y la temporización relativa a una secuencia finita de mensajes.
Rechazo	Negación por parte de alguna de las entidades participantes en una comunicación.
Rede de computadoras	Colección interconectada de computadoras autónomas
Reino	Dominio de autentificación de Keberos
RSA	Criptosistema con clave pública (*Rivest*, *Shamir* y *Adleman*).
Servidor	Computadora que contiene una configuración necesaria para ser reconocido como parte de la red Internet. Adicionalmente, se llama así a los sistemas que proporcionan recursos, como servidores de archivos y de nombres, y resuelve las peticiones emanadas desde los programas llamados clientes. Un

	servidor también es aquella computadora que contiene dichos programas.
SESAME	Sistema de autentificación y distribución de claves que esta siendo desarrollado como parte de un proyecto de investigación y desarrollo.
Sistema	Entidad direccionable dentro de una red de computadoras o sistemas distribuidos. La entidad se direcciona generalmente bien por su nombre o por su dirección de capa de red.
Sistemas distribuido	Red de computadoras, que dispone de un esquema cliente-servidor, se denomina cliente a la computadora que solicita un determinado servicio y se denomina servidor a la computadora que lo proporciona, dicho servicio puede ser la ejecución de un determinado algoritmo, el acceso a determinado banco de información o el acceso a un dispositivo hardware o software. Por extensión, se puede aplicar el esquema cliente-servidor dentro de una misma máquina, donde el proceso servidor y el proceso cliente son dos procesos independientes que corren dentro de la misma instancia de sistema operativo siendo transparente y por tanto no es necesariamente visible para el usuario.
Sniffer	Literalmente "Husmeador". Pequeño programa que busca una cadena numérica o de caracteres en los paquetes que atraviesan un nodo con objeto de conseguir alguna información. Normalmente su uso es ilegal.
SPX	Sistema de autentificación y distribución de claves y prototipado por DEC.
SSL	*Secure Socket Layer* - Capa de *Socket* Segura. Protocolo que ofrece funciones de seguridad a nivel de la capa de transporte para TCP. Es el conjunto de especificaciones, donde el *browser* del cliente encripta la información antes de enviarla, haciendo que sólo un receptor legítimo pueda descifrar el mensaje.
TESS	Sistema formado por conjunto de herramientas de mecanismos y funciones criptográficas diferentes que

	cooperan y que están basadas en primitivas de exponenciación discreta.
TGS	*Ticket Granting Server* - Servidor de concesión de *tickets*.
TGT	*Ticket Granting Ticket* - *Ticket* de concesión de *tickets*.
Ticket	Credencial electrónica que contiene datos que son utilizados para la autentificación.
UNIX 4.3 BSD	El UNIX 4.3 BSD (*Berkeley Standard Distribution*), es un sistema operativo desarrollado por la Universidad de California en *Berkeley*.
Valor Temporal	Número aleatorio reciente y no predecible.

ÍNDICE DE FIGURAS

Pág.

ÍNDICE DE TABLAS

Pág.

BIBLIOGRAFÍA

[ANGEL,1999] "Criptografía para principiantes", José de Jesús Ángel Ángel, SeguriDATA, Septiembre 1999.

[OPPLIEGER, 1998] "Sistemas de Autentificación para Seguridad en Redes", Rolf Oppliger, AlLFAOMEGA, Bogota Colombia, Junio 1998.

[CHESWICK;BELLOVIN, 1994] "Firewalls And Internet Security: Repelling the Wily Hacker", W.R. Cheswick and S.M. Bellovin, Addison- Wesley, Massachusets, 1994.

[CABALLERO, 1996] Introducción a la Criptografía, Segunda Edición, P. Caballero, Alfaomega - Rama, 1996.

[ROBLINNG, 1982] "Cryptography and Data Scurity", Dorthy Elizabeth Roblinng Denning, Addison – Wesley Publishing Company, Massachusetts, 1982.

[MAYEL;MATYAS,1982] "Cryptography: A New Demension in Computer Data Security", Carl H. Meyer and Stephen M. Matyas, John Wiley & Sons, New York, 1982 .

[HALSALL, 1998] "Comunicaciones de datos, redes de computadores y sistemas abiertos", Fred Halsall, Person Education, 1998.

[FROUFE,1997] "Técnicas criptográficas de protección de datos", Agustín Froufe, ALFAOMEGA, 1997.

[TANENBAUN,1997a] "Distributed Operating Systems -- Concepts & Practices", Adrew S. Tenenbaun, Prentice Hall, 1997.

[BLACK, 1995] Segunda Edición "Redes de Computadores: Protocolos, Normas e Interfaces", Uyless Black, Ramma, 1995.

[TANENBAUN,1997b] Tercera Edición, "Redes de Computadores", Andrew S. Tenenbaum, Prentice-Hall, 1997.

[LAM; BHET, 1992] "Timely Authentication in Distributed System" K. Y. Lam and T. Bhet, Springer – Verlag, Berlin, Noviembre 1992.

[COULOURIS, et al., 1994] Second Edition "DISTRIBUTED SYSTEMS: CONCEPTS AND DESIGN", George Coulouris, Jean Dollimore and Tim Kindberg, Addison-Wesley Publishing Company, 1994.

[BELLOVIN;MERRITT,1990] "Limitations of the Kerberos Authentication System", Steven M. Bellovin , Michael Merritt, ACM Computer Communication, 1990.

[PFAFFENBERGER;WALL,1996] Sexta Edición, "Diccionario Para Usuarios de Computadoras e Internet, Bryan Pfaffenberger, David Wall, Prentice Hall, 1996.

DIRECCIONES ELECTRÓNICAS

[1] http://www.latinoseguridad.com/Reps/Cripto1.htm

[2] http://ieee.udistrital.edu.co/concurso/ciencia_tecnologia_info_3/introduccion_1.html#utilidades

[3] http://leo.worldonline.es/jlquijad/histo.htm#II.%20MÉTODOS

[4] http://www.isaca.org/percep/v2art4.htm

[5] http://www.linuxfocus.org/Castellano/May2002/article243.shtml

[6] http://www.camerfirma.com/mod_web/usuarios/faqpg.html

[7] http://www.htmlweb.net/seguridad/cripto/cripto_3.html

[8] http://www.ciberia.ya.com/rvalle2001/index.html

[9] http://www.coit.es/publicac/publbit/bit128/bitcd1/htm/herramientas/unix¬sec -1.2/node22.html

[10] http://www.rediris.es/cert/doc/unixsec/node29.html

[11] http://www.eumed.net/cursecon/ecoinet/seguridad/asimetrica.htm

[12] http://web.mit.edu/kerberos

[13] http://penta.ufrgs.br/gereseg/unlp/t10home.htm

[14] http://www.itq.edu.mx/vidatec/espacio/aisc/kerberos/dialogo.htm

[15] http://web.mit.edu/kerberos/www/dialogue.html

[16] http://www.itq.edu.mx/vidatec/espacio/aisc/kerberos/#Escena

[17] http://asignaturas.diatel.upm.es/seguridad/autenticacion.htm

[18] http://ditec.um.es/laso/docs/tut-tcpip/

[19] http://web.mit.edu/kerberos/www

[20] http://dimacs.rutgers.edu/Workshops/Security/program2/boyd/node14.html

[21] http://dimacs.rutgers.edu/Workshops/Security/program2/focardi/node6.html

[22] http://www.objs.com/survey/encrypt.htm

[23] http://www.rediris.es/cert/doc/unixsec/node27.html

[24] http://www.iespana.es/mundolinux/linux/unixsec/node20.html

[25] http://penta.ufrgs.br/gereseg/unlp/t10home.htm

[26] http://www.pasta.cs.uit.no/thesis/html/ronnya/node41.html

[27] http://www.microsoft.com/resources/documentation/windowsServ/2003¬
/all/techref/en-us/w2k3tr_kerb_how.asp

[28] http://www.dccia.ua.es/dccia/inf/asignaturas/TIC/Material/Intercambio¬
%20de%20claves/P5.9.htm

OTRAS PUBLICACIONES

[RED, 1995]	La Revista de Redes de Computadora, Noviembre 1995, "¿Necesita una fiera para proteger su información?", Ing. en Sist. Comp. .Georgina Castañón, Pág. 4 – 8.