Programación Web
Unidad 1

Introducción a la Web y HTTP

Fundamentos de la Web

Los Fundamentos de la Web se refieren a los principios básicos que sustentan el funcionamiento de Internet y las tecnologías asociadas que permiten la navegación, interacción y acceso a información a través de la World Wide Web. Estos fundamentos comprenden varios conceptos clave, entre los que se incluyen:

- **Protocolo HTTP (Hypertext Transfer Protocol):** Es el protocolo de comunicación utilizado para transferir datos en la web. Define cómo se envían y reciben las solicitudes y respuestas entre los clientes (navegadores web) y los servidores web.

- **URI (Uniform Resource Identifier):** Es una cadena de caracteres que identifica de manera única un recurso en la web, como una página web, una imagen o un archivo. Las URLs (Uniform Resource Locators) son un tipo de URI que especifica la ubicación de un recurso en Internet.

- **HTML (Hypertext Markup Language):** Es el lenguaje de marcado utilizado para crear páginas web. Define la estructura y el contenido de una página mediante etiquetas y elementos que describen el texto, imágenes, enlaces y otros elementos presentes en la página.

- **CSS (Cascading Style Sheets):** Es un lenguaje utilizado para dar estilo y diseño a las páginas web creadas con HTML. Permite controlar la apariencia visual de los elementos HTML, como el color, el tamaño, la tipografía y la disposición en la página.

- **JavaScript:** Es un lenguaje de programación utilizado para agregar interactividad y dinamismo a las páginas web. Permite manipular el contenido de la página en tiempo real, responder a eventos del usuario y comunicarse con servidores web para cargar datos adicionales sin recargar la página completa.

- **DOM (Document Object Model):** Es una representación en memoria de la estructura de una página web creada con HTML. Permite acceder y manipular los elementos de la página mediante programación, lo que facilita la interacción dinámica con el contenido de la página.

¿Cómo se comunican las máquinas?.

Para poder analizar el tráfico de una red necesitamos conocer previamente cómo se comunican las máquinas, aquí tenemos que hablar de los protocolos, del modelo OSI, de las unidades de protocolo de datos (PDU) y del hardware necesario para capturar este tráfico.

Las redes de ordenadores están formadas por diferentes sistemas corriendo en diferentes plataformas. Para poder comunicarse entre ellas utilizan una serie de lenguajes o reglas que llamamos protocolos. Entre los protocolos más comunes están: **TCP** (Transmission Control Protocol), **IP** (Internet Protocol), **ARP** (Address Resolution Protocol) o **HTTP** (Hypertext Transfer Protocol). Cuando tenemos un conjunto de protocolos que trabajan juntos se llama «protocol stack» o pila de protocolos.

Podemos pensar tanto en los protocolos como en el lenguaje que utilizamos las personas para comunicarnos. Tenemos reglas para conjugar los verbos, para saludar a la gente, o incluso para dar las gracias. Pues los protocolos actúan de la misma forma, definiendo como «enrutar» o encaminar los paquetes, cómo iniciar una conexión, cómo confirmar la recepción de los datos, etc…

Los protocolos pueden ser simples o complicados dependiendo de su función. Sin embargo, la mayor parte de los protocolos tienen que lidiar con los siguientes problemas:

• **Inicio de la conexión**: ¿Quién inicia la conexión, el cliente o el servidor? ¿Qué información debe ser intercambiada antes de iniciar la comunicación?
• **Negociación de las características de la conexión**: ¿La comunicación es cifrada? ¿Cómo transmitimos las «encryption keys» entre las máquinas?
• **Formato de los datos**: ¿Cómo organizamos el contenido de los datos de la comunicación en el paquete? ¿En qué orden se procesan los datos al recibirlos?
• **Detección y corrección de errores**: ¿Qué ocurre si el paquete tarda mucho tiempo en llegar a su destino? ¿Qué ocurre si el cliente pierde la conexión con el servidor durante un breve espacio de tiempo?
• **Fin de la conexión**: ¿Cómo le indica una máquina a la otra que la conversación ha terminado? ¿Qué información debemos enviar para terminar una conexión de forma correcta?

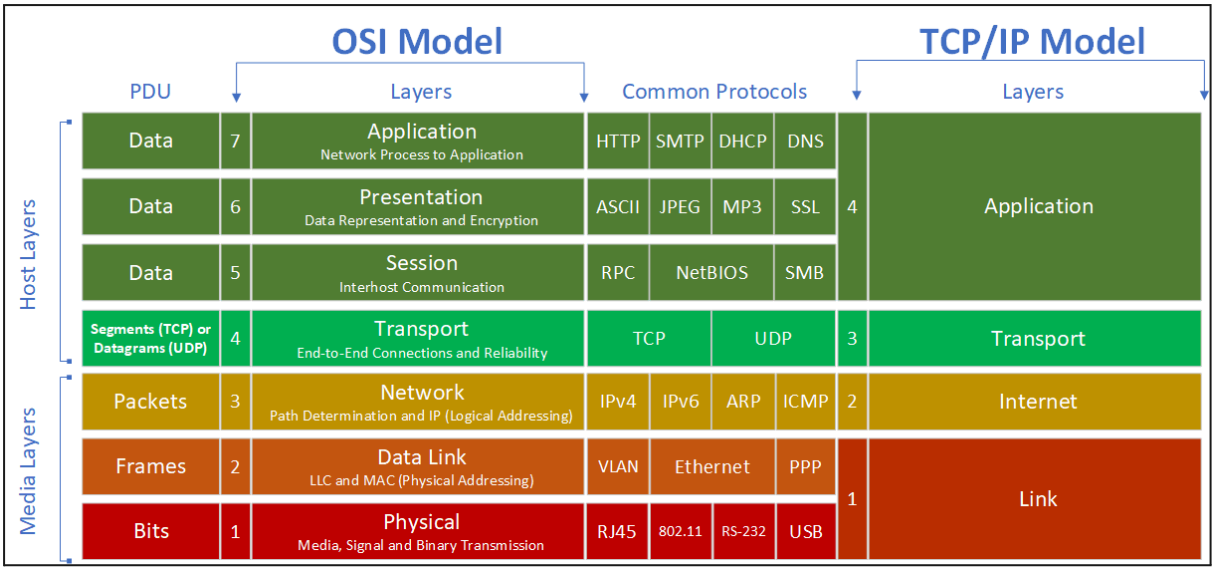
El modelo OSI

El modelo OSI (Open System Interconnection) o también llamado la pila de capas o niveles del modelo OSI, es un modelo de referencia ideal para los protocolos de la red, los cuales están separados de acuerdo a sus funciones. Fue desarrollado originalmente en 1983 por la International Organization for Standardization (ISO) en un documento llamado ISO 7498. Muchos de los estándares están disponibles mediante pago pero algunos pueden ser libremente descargados desde aquí. Es simplemente un estándar, una recomendación para la industria. Los desarrolladores de software o hardware no tienen por qué seguirlo al pie de la letra. De hecho, ***no lo siguen***, ya que el más utilizado es el modelo **TCP/IP**.

En el modelo OSI tenemos **siete capas** o niveles distintos («layers» en inglés) que nos permiten entender perfectamente las comunicaciones de red:

• Las 4 capas de la parte de arriba se llaman «Host layers» y son las responsables de proporcionar una entrega precisa de los datos entre los equipos.
• Las 3 capas de la parte de abajo se llaman «Media layers» y son las responsables de controlar la entrega física de los datos a través de la red.

Las funciones de cada capa, nivel o «layer» son las siguientes:



– **Application (layer 7)**: La capa de aplicación es el nivel más alto del modelo OSI, es la capa vista por los usuarios finales, proporciona la interfaz de comunicación entre las aplicaciones y los distintos servicios de red. Aquí se encuentran los protocolos HTTP, SMTP, FTP, DHCP, DNS, etc…

– **Presentation (layer 6)**: La capa de presentación se encarga de transformar los datos que recibe en un formato que pueda ser leído por la capa de aplicación o por las capas inferiores. La codificación o decodificación hecha de los datos, depende del protocolo de aplicación que envía o recibe los datos. Realiza funciones de compresión de datos. Aquí también tiene lugar el cifrado o descifrado para segurizar las comunicaciones. Ejemplos de protocolos de esta capa son ASCII, JPEG, MP3, TLS, SSL, etc…

– **Session (layer 5)**: La capa de sesión controla el diálogo (o sesión, valga la redundancia) entre los dispositivos. Establece, gestiona, mantiene y termina las conexiones entre los equipos involucrados en la comunicación. Es responsable de establecer si la comunicación es full-duplex (simultáneamente en los dos sentidos) o half-duplex (en un solo sentido al mismo tiempo) y se encarga de terminar correctamente una conexión entre dispositivos. También tienen lugar aquí las funciones de autenticación y autorización. Ejemplos de protocolos de sesión son RPC, NetBIOS, SMB, etc…

– **Transport (layer 4)**: La capa de transporte tiene como principal objetivo garantizar la correcta transferencia de los datos entre equipos manteniendo la calidad del servicio y asegurando que la comunicación se produzca libre de errores. Esto se consigue vía mecanismos de corrección de errores, con segmentación o desegmentación, con controles de flujo, etc… El transporte de los datos es algo muy importante y un tema bastante extenso, y es por eso que el modelo OSI le dedica una capa entera. Aquí tenemos principalmente dos protocolos: TCP (orientado a conexiones o «connection-oriented») y UDP (no orientado a conexiones o «connectionless»). Algunos firewalls y servidores proxy operan en esta capa.

– **Network (layer 3)**: La capa de red es una de las capas más complejas del modelo OSI, es responsable de enrutar los datos entre las distintas redes físicas. Utiliza las direcciones lógicas de los dispositivos de red (por ejemplo, las direcciones IP). Se encarga también, si es necesario, de dividir los datos en fragmentos más pequeños («splitting») y en algunos casos de la detección de errores. Los routers operan en esta capa. Son protocolos de esta capa IPv4 e IPv6, ARP, ICMP, etc…

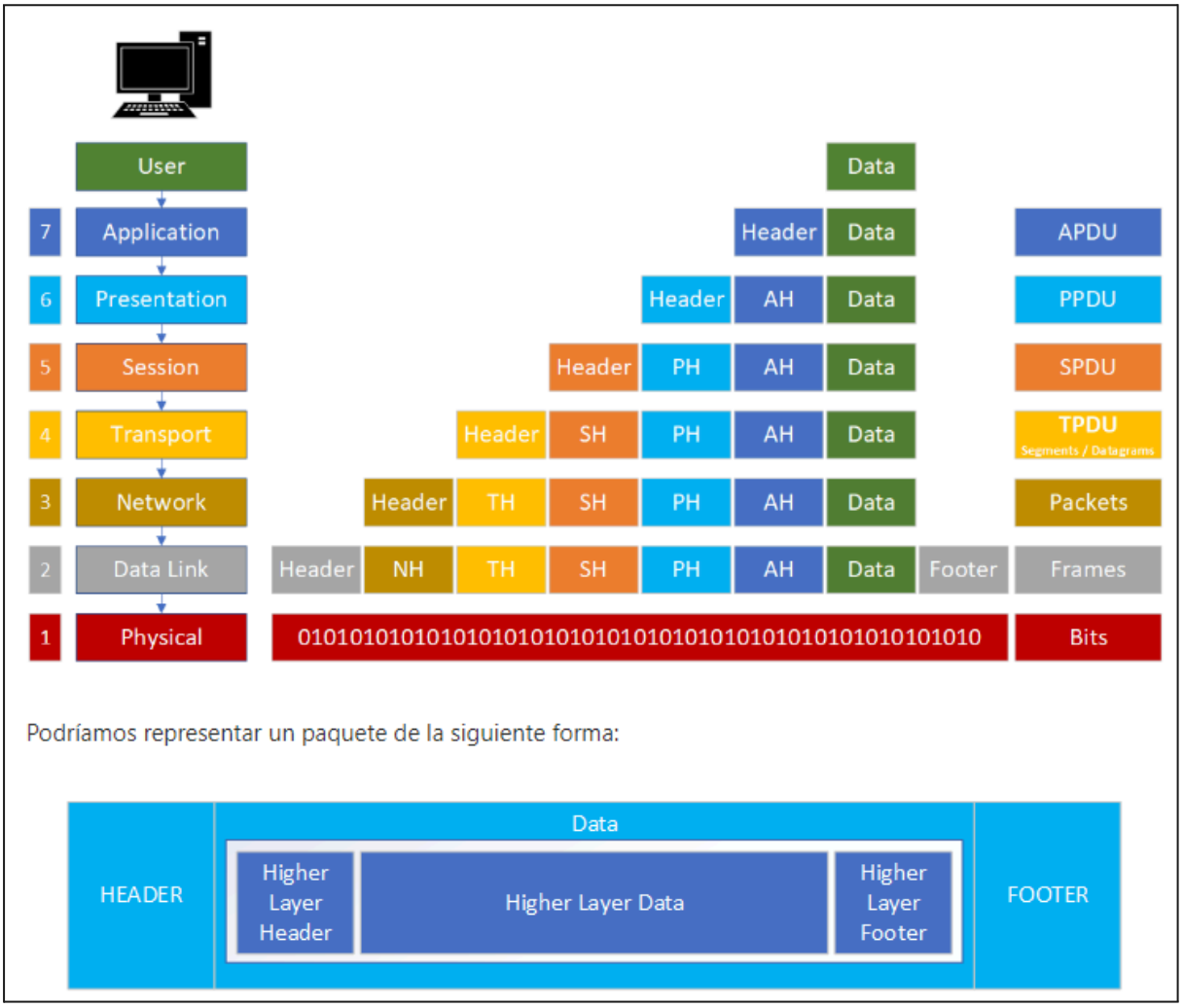
– **Data link (layer 2)**: La capa de enlace proporciona la transferencia de datos de nodo a nodo, un enlace o medio de transporte entre dos o más dispositivos directamente conectados. Su principal objetivo es realizar un esquema de direcciones para identificar los dispositivos (por ejemplo, a través de las MAC). Generalmente se divide en dos subcapas: una superior llamada LLC (Logical Link Control) y una inferior llamada MAC (Media Access Control). LLC define la forma de transferencia de los datos mientras que MAC se encarga del direccionamiento, por poner un ejemplo, la electrónica que utiliza una tarjeta de red (NIC) para comunicarse con otras tarjetas de red sería la subcapa MAC, mientras que el controlador o driver sería la subcapa LLC. Los switches operan en esta capa. Ojo, hablamos de switches de capa 2, existen también switches de capa 3 que operan también en la capa de red (capaces de trabajar con direcciones MAC e IP). Los Protocolos de esta capa son Ethernet, Token Ring, PPP, VLAN, etc…

– **Physical (layer 1)**: La capa física es la más baja del modelo OSI y es el medio físico por el cual los datos de red son transferidos. Establece una conexión física entre los dispositivos para habilitar la comunicación entre ellos. La información que se intercambia aquí se transfiere en bits (unos y ceros). En esta capa se define el hardware y las características eléctricas empleadas, incluyendo voltajes, conectores, tipos de cable, pines, tarjetas de red, repetidores, etc… Esta capa define la topología de la red, establece y termina conexiones y convierte también señales digitales en analógicas (y viceversa). Los Protocolos de esta capa incluyen todos los relacionados con el cableado o las conexiones inalámbricas como pueden ser RJ45, 802.11, RS-232, USB, DSL, etc…

Encapsulación de los datos

Los protocolos de las distintas capas del modelo OSI intercambian los datos entre ellas con la ayuda de la **encapsulación**. Cada capa es responsable de añadir una cabecera (header) o cola (footer o trailer) a los datos transferidos. Estas headers o footers no son más que unos bits extra de información que permiten a las capas comunicarse entre ellas.

El proceso de la encapsulación da como resultado una **unidad de protocolo de datos o PDU** (Protocol Data Unit), el cual incluye los datos que se envían y además todas las headers o footers añadidos.



Introducción a la World Wide Web.

La World Wide Web (WWW), también conocida simplemente como la web, es un vasto universo digital que ha transformado la forma en que interactuamos, compartimos información y nos comunicamos a nivel global. Desde su creación a principios de la década de 1990 por Tim Berners-Lee, la web ha evolucionado de ser una colección estática de documentos interconectados a una red dinámica y colaborativa que abarca todos los aspectos de la vida moderna.

En su esencia, **la web se compone de una intrincada red de servidores y clientes interconectados que se comunican a través del Protocolo de Transferencia de Hipertexto (HTTP)**. Este protocolo permite a los usuarios acceder y navegar por una amplia variedad de recursos en línea, como páginas web, imágenes, videos, documentos y aplicaciones, a través de navegadores web como Chrome, Firefox y Safari.

Lo que hace que la web sea tan poderosa es su capacidad para democratizar el acceso a la información y fomentar la colaboración y la creatividad a escala global. Desde la publicación de contenido personal en blogs y redes sociales hasta la creación de aplicaciones web innovadoras y el comercio electrónico, la web ofrece un espacio sin límites para la expresión, el intercambio de ideas y el desarrollo de proyectos colaborativos.

Además, la web es una fuente inagotable de conocimiento y recursos educativos. Desde cursos en línea y tutoriales hasta bibliotecas digitales y bases de datos de investigación, la web ofrece acceso a una gran cantidad de información que antes estaba fuera del alcance de muchas personas.

Sin embargo, a medida que la web continúa expandiéndose y evolucionando, también enfrenta desafíos y preocupaciones relacionadas con la privacidad, la seguridad y la desinformación. Es importante abordar estos problemas de manera proactiva para garantizar que la web siga siendo un espacio seguro y accesible para todos.

Breve historia de la Web

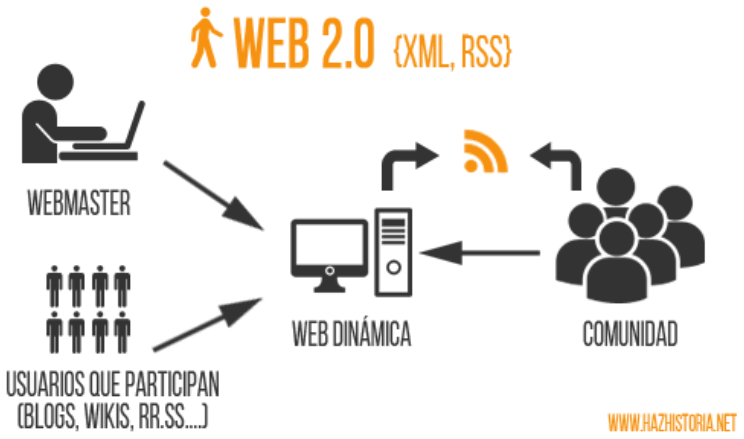
La historia de la World Wide Web (WWW) es un relato fascinante que abarca décadas de innovación, colaboración y evolución tecnológica. Desde sus modestos comienzos en la década de 1960 hasta su estado actual como una red global de información y comunicación, la web ha experimentado varias generaciones, cada una marcada por avances significativos en la tecnología y la forma en que interactuamos con ella.

La primera generación de la web se remonta a los años 60 y 70, cuando visionarios como Ted Nelson y Douglas Engelbart imaginaron un *sistema de hipertexto que permitiría a los usuarios navegar por una red de documentos interconectados*. Estos conceptos se materializaron en proyectos como el sistema Xanadu de Nelson y el famoso demo de Engelbart en 1968, que introdujo el concepto de ratón y ventanas en la informática.

Se trataba de un servicio de comunicación que sólo permitía la inclusión de textos y eran manipulables a través de navegadores de sólo texto. Sin embargo, no fue hasta principios de los noventa cuando se creó HTML, lo que provocó que la web empezara a tener una aceptación suficiente y extenderse como la pólvora. En aquel entonces, *la web era un sistema unidireccional de publicación estático de sólo texto que no presentaba gráficos o*

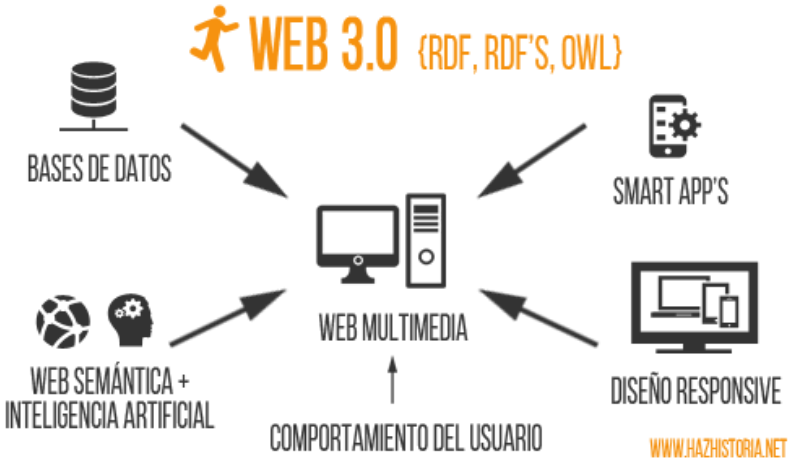
imágenes, no ofrecía opciones de personalización, no permitía la actualización y, mucho menos, la posibilidad de realizar intercambio de datos, por lo que los usuarios no podían interactuar con el contenido y, únicamente, se limitaban a consultar o leer la información que el administrador de la página web hubiese subido a la red. A esto se denominó la **Web 1.0.**

La **segunda generación** de la web surgió en la década de 1980 con el desarrollo de protocolos de comunicación como TCP/IP y la creación de Arpanet, precursora de Internet. Tim Berners-Lee, un científico de la computación en el CERN, desarrolló el concepto de la World Wide Web en 1989 como una forma de compartir y acceder a información en línea mediante hipervínculos y navegadores web. En 1991, Berners-Lee lanzó el primer sitio web y el primer navegador web, dando inicio a la era de la web moderna.



La **web 2.0**, término que fue bautizado por O'Reilly en el año 2004, supuso la segunda generación de sitios web y páginas web. Este tipo de webs ya no eran estáticas, ni de sólo lectura y permitían, entre otras cosas, *compartir e interactuar con la información de una manera sencilla*. Como consecuencia de ello, se produjo un desarrollo de la inteligencia colectiva que fomentaba la colaboración y el intercambio de información a través de comunidades y redes sociales. Es, por esta época, cuando se crean y extienden sistemas tan conocidos como son los blogs, chats, wikis o foros. Sistemas bidireccionales, los cuales, permitían manipular y gestionar la información de forma sencilla, además de permitir la adición de comentarios y opiniones o interactuar con otros usuarios que presentaban las mismas inquietudes, pero que no requerían tener el mismo nivel técnico o cultural.

La **tercera generación** de la web se caracterizó por el rápido crecimiento de la cantidad de sitios web y la popularización de la web entre el público en general. Marcas como Netscape Navigator y Internet Explorer se convirtieron en los navegadores dominantes, y surgieron servicios como Yahoo! y Amazon.com, que introdujeron la búsqueda y el comercio electrónico en línea.



La **web 3.0**, la tercera generación de sitios y páginas web, ya no sólo era una forma de interactuar y compartir la información de manera sencilla, ahora, su objetivo es darle significado y enriquecer la experiencia del usuario. Es aquí, como alguno ya habrá pensado, cuando nace la Semántica web y las páginas web empiezan a estructurarse a través de un lenguaje natural que puede ser interpretado por el software definiendo qué parte tiene cada función. De esta forma, acceder a la información resulta más sencillo y rápido de procesar, porque hasta las máquinas son capaces de “entender” los contenidos y su objetivo. Se dice que la web 3.0 también tiene bastante que ver con la inteligencia artificial puesto que las páginas y aplicaciones web ya poseen la capacidad de conectarse entre sí para ofrecer un mejor servicio a los intereses de cada usuario.

La **cuarta generación** de la web, a principios del siglo XXI, se centró en la interactividad y la participación del usuario. Plataformas como Google, Facebook y YouTube transformaron la web en un espacio social y colaborativo, permitiendo a los usuarios compartir contenido, conectarse con otros y contribuir a la creación de la web.

Con la **web 4.0**, nacen el aprendizaje profundo (Deep Learning) y el aprendizaje automático (Machine Learning), tecnologías que forman parte de una familia de métodos de aprendizaje automático basados en redes neuronales con aprendizaje de representación. En otras palabras, tecnologías basadas en sistemas capaces de aprender a realizar tareas tras analizar diferentes patrones y muestras mediante técnicas de aprendizaje que permiten descubrir de manera automática las características de una entidad a partir de datos sin procesar. El ejemplo más conocido o extendido de todo esto quizás sea Watson de IBM, un software capaz de responder preguntas realizadas en lenguaje natural y de realizar tareas como Speech To Text, el cual permite hablarle a una máquina y convertir lo dicho en texto escrito. Pero esto no es todo, la web 4.0 es la responsable de que los usuarios sean advertidos por sus dispositivos móviles antes de que ellos mismos se den cuenta. Por ejemplo, ¿quién no ha recibido notificaciones con la ruta más corta al trabajo, avisos por atascos en la carretera, alertas por fuertes tormentas o lluvias o mensajes de advertencia sobre tu elevado ritmo cardíaco?

La **quinta generación** de la web, que estamos experimentando en la actualidad, se caracteriza por la convergencia de la web y las tecnologías emergentes como la inteligencia artificial, el Internet de las cosas (IoT) y la realidad aumentada. Esto está dando lugar a nuevas experiencias de usuario, como la búsqueda por voz, los asistentes virtuales y los dispositivos conectados, que están redefiniendo la forma en que interactuamos con la web y el mundo que nos rodea.

Arquitectura Cliente-Servidor y Protocolo HTTP

La **Arquitectura Cliente-Servidor** es un modelo de diseño de software que distribuye las funciones de una aplicación entre dos tipos de entidades: el cliente y el servidor. Esta arquitectura establece una comunicación estructurada y eficiente entre estos dos componentes, permitiendo que los usuarios accedan y manipulen datos y recursos de manera remota a través de una red.

En este modelo, el cliente es la interfaz de usuario o la aplicación que solicita y consume los servicios proporcionados por el servidor. El cliente puede ser un programa de software instalado en una computadora, **un navegador web** en un dispositivo móvil o incluso un dispositivo físico como un sensor IoT. Su función principal es enviar solicitudes al servidor y mostrar los resultados al usuario de una manera comprensible.

Por otro lado, el servidor es el **componente que almacena, procesa y gestiona los datos y recursos de la aplicación**. Es responsable de recibir las solicitudes del cliente, procesarlas y enviar las respuestas adecuadas de vuelta al cliente. El servidor generalmente se ejecuta en una computadora dedicada con una mayor capacidad de procesamiento y almacenamiento, lo que le permite manejar múltiples solicitudes de clientes simultáneamente.

La comunicación entre el cliente y el servidor se basa en un conjunto de protocolos de red estándar, como HTTP para aplicaciones web o TCP/IP para aplicaciones más genéricas. Esta comunicación puede ser **síncrona**, donde el cliente espera una respuesta inmediata

del servidor, o **asíncrona**, donde el cliente continúa su ejecución mientras espera la respuesta del servidor en segundo plano.

La Arquitectura Cliente-Servidor ofrece varias ventajas, como la modularidad, la escalabilidad y la capacidad de distribuir la carga de trabajo entre múltiples servidores. Además, facilita la implementación de actualizaciones y mejoras en la aplicación, ya que los cambios realizados en el servidor se reflejan automáticamente en todos los clientes.

De forma básica, cuando un usuario se conecta a Internet con un dispositivo cualquiera, se le asigna un identificador único mediante los protocolos TCP/IP (Protocolo de Control de Transmisión / Protocolo de Internet). El protocolo TCP proporciona el medio para crear las conexiones y el protocolo IP proporciona el mejor “camino” para alcanzar su destino.

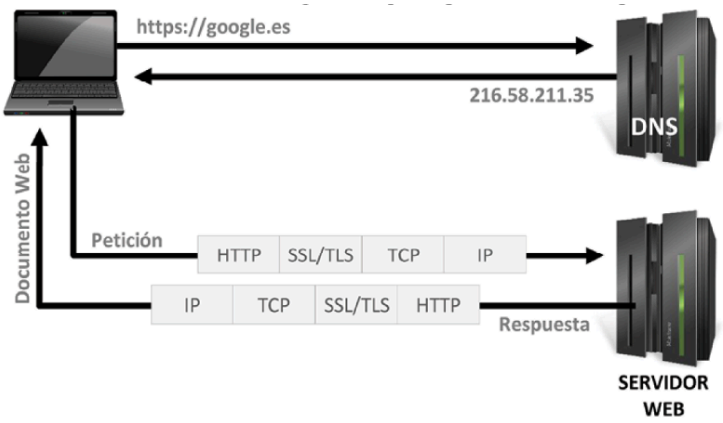
Este identificador único, más conocido como dirección IP, suele estar compuesto por cuatro códigos de 8 bits y vinculado a un nombre, también único, el cual utilizamos para acceder a un sitio web (véase, por ejemplo, https://google.es).

¿Qué es lo que sucede entre medias? Como hemos dicho, Internet se mueve a través de direcciones IP, por lo que, para conseguir la dirección IP asignada a ese nombre que hemos introducido, primero se debe acceder a un **sistema intermedio que almacena dicha relación**.

Ese sistema intermedio se conoce como **DNS** (Sistema de Nombres de Dominio) y, fundamentalmente, lo que hace es recopilar un catálogo de correspondencias de nombres e IPs y devolver un valor concreto como, por ejemplo, 216.58.211.35.

Una vez que se tiene el objetivo al que dirigirse, el navegador, también llamado Cliente en términos de comunicaciones, abre una instancia de comunicación con el Servidor mediante el protocolo **HTTP (Protocolo de Transferencia de Hipertexto).** Este protocolo es quién dicta las normas para que el Cliente se comunique con el Servidor Web asignado a la IP anteriormente adquirida y es, además, quién define la sintaxis y semántica que se debe utilizar en cada conexión.

No obstante, si accedemos a la consola del navegador (pulsando F12) y recuperamos la información de la pestaña NETWORK, al recargar la página veremos que la mayoría de estas conexiones entre el Cliente y el Servidor se realizan a través de **HTTPS**, o lo que es lo mismo, la versión segura del protocolo HTTP.



En este tipo de comunicación, el servidor establece un cifrado basado en la seguridad de textos mediante los protocolos criptográficos SSL/TLS, los cuales, permiten crear una capa codificada intermedia entre los protocolos HTTP y TCP/IP por el que envía el código HTML que el navegador muestra al usuario.

Protocolo HTTP: métodos, códigos de estado y cabeceras

El **Protocolo de Transferencia de Hipertexto (HTTP)** es el protocolo de comunicación utilizado para la transferencia de datos en la World Wide Web. Está basado en un modelo cliente-servidor, donde un cliente, típicamente un navegador web, realiza solicitudes a un servidor web para acceder a recursos como páginas web, imágenes, videos, entre otros.

El protocolo HTTP define una serie de métodos que especifican la acción que el cliente desea realizar sobre un recurso determinado en el servidor. Algunos de los **métodos** más comunes incluyen:

- **GET**: Solicita la recuperación de un recurso específico del servidor. Es utilizado para obtener información del servidor, como páginas web, imágenes o archivos.

- **POST**: Envía datos al servidor para que sean procesados. Es utilizado comúnmente en formularios web para enviar datos al servidor, como información de registro o datos de un formulario.

- **PUT**: Envía datos al servidor para ser almacenados en un recurso específico. Es utilizado para actualizar o crear recursos en el servidor.

- **DELETE**: Solicita la eliminación de un recurso específico en el servidor. Es utilizado para eliminar recursos almacenados en el servidor.

Además de los métodos, el protocolo HTTP define **códigos de estado** que indican el resultado de la solicitud realizada por el cliente. Algunos de los códigos de estado más comunes son:

- **200 OK:** Indica que la solicitud se ha completado con éxito y que el servidor ha devuelto los datos solicitados al cliente.

- **404 Not Found:** Indica que el recurso solicitado no se ha encontrado en el servidor. Es comúnmente utilizado cuando una página web o recurso no está disponible.

- **500 Internal Server Error:** Indica que se ha producido un error interno en el servidor al procesar la solicitud del cliente. Es utilizado cuando el servidor encuentra un problema que impide completar la solicitud del cliente.

Por último, el protocolo HTTP también permite el uso de cabeceras HTTP para proporcionar información adicional sobre la solicitud o la respuesta.

Las cabeceras HTTP son componentes clave de las solicitudes y respuestas HTTP que se intercambian entre clientes y servidores en la World Wide Web. Estas cabeceras proporcionan información adicional sobre la solicitud o la respuesta, lo que permite una comunicación más completa y eficiente entre los componentes de la aplicación web.

Tipos de Cabeceras HTTP:

- **Cabeceras de Solicitud**: Enviadas por el cliente al servidor para proporcionar información sobre la solicitud, como el método HTTP utilizado, la URL solicitada, el tipo de contenido aceptado por el cliente, las cookies, etc.
- **Cabeceras de Respuesta**: Enviadas por el servidor al cliente para proporcionar información sobre la respuesta, como el código de estado HTTP, el tipo de contenido devuelto, la longitud del contenido, las cookies, etc.

Funciones de las Cabeceras HTTP:

- **Control de Caché**: Las cabeceras como Cache-Control, Expires y ETag permiten al servidor controlar cómo los recursos son almacenados en caché por los clientes, lo que puede mejorar el rendimiento y reducir el consumo de ancho de banda.
- **Negociación de Contenido**: Las cabeceras como Accept y Content-Type permiten a los clientes y servidores negociar el tipo de contenido que se enviará o recibirá en la solicitud y la respuesta, respectivamente.
- **Seguridad**: Las cabeceras como X-Frame-Options, Content-Security-Policy y Strict-Transport-Security permiten al servidor especificar políticas de seguridad para protegerse contra ataques como el clickjacking, la ejecución de scripts no autorizados y los ataques de intercepción de red.
- **Autenticación y Autorización**: Las cabeceras como Authorization permiten a los clientes autenticarse en el servidor, mientras que las cabeceras como WWW-Authenticate permiten al servidor solicitar credenciales de autenticación al cliente.
- **Control de Cookies**: Las cabeceras como Set-Cookie y Cookie permiten al servidor enviar y recibir cookies para realizar un seguimiento del estado de la sesión del usuario y personalizar la experiencia del usuario en el sitio web.

La seguridad y el rendimiento en internet son de vital importancia, lo que hace que comprender la diferencia entre **HTTP y HTTPS** sea algo esencial para cualquier usuario de internet. HTTP es conocido por haberse convertido en el estándar para la transferencia de datos en la web, aunque de unos años a esta parte HTTPS ha emergido como una versión más segura y eficiente.

HTTPS, que significa Hypertext Transfer Protocol Secure, es la versión segura de HTTP. ¿En qué consiste el protocolo HTTPS? Incorpora una capa adicional de seguridad mediante el uso de certificados SSL (Secure Socket Layer) o TLS (Transport Layer Security) para cifrar las comunicaciones entre el navegador del usuario y el servidor web.

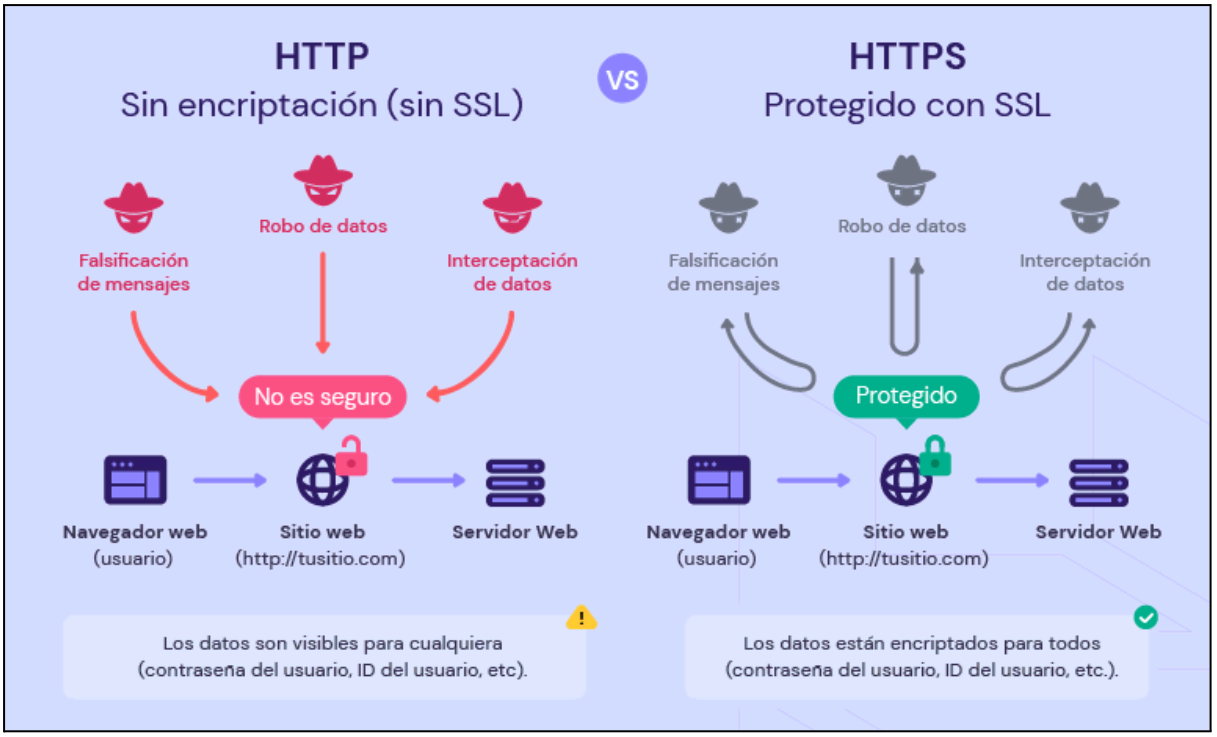
Esto asegura que cualquier dato transferido, como información personal o detalles de tarjetas de crédito, permanezca privado y protegido contra posibles incursiones externas con malas intenciones.

Además de mejorar la seguridad de una web, HTTPS también autentifica las páginas, asegurando a los usuarios que están conectados a la web correcta y no a una réplica fraudulenta. Con la creciente preocupación en la seguridad en línea, HTTPS se está

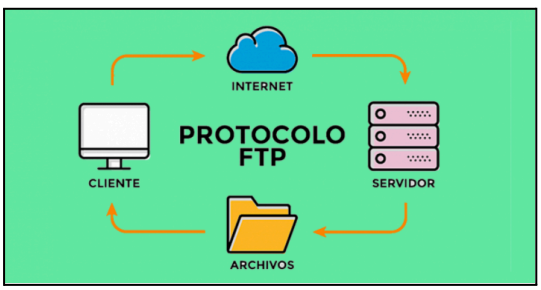
convirtiendo rápidamente en el estándar para todas las páginas web independientemente de su finalidad o su temática.

El protocolo HTTPS es fundamental para la seguridad de las páginas web. Al cifrar los datos transferidos entre el navegador y el servidor, HTTPS protege la información sensible de los usuarios, como contraseñas y detalles de tarjetas de crédito, contra posibles ciberataques e interceptaciones de terceros.

Esta capa de seguridad adicional es imprescindible **no solo para webs de comercio electrónico y bancos en línea, sino para cualquier página que maneje datos personales**, como por ejemplo a través de la inclusión de formularios de Google que pretendan generar una buena base de leads. Además, en esta batalla de HTTP vs HTTPS, hay que decir que el segundo mejora la confianza de los usuarios en la página web, lo que es esencial para mantener la reputación y la credibilidad en el entorno digital.



Protocolo FTP y SSL



El **Protocolo de Transferencia de Archivos** (FTP, por sus siglas en inglés File Transfer Protocol) es un protocolo estándar utilizado para la transferencia de archivos entre un cliente y un servidor en una red de computadoras, como Internet. Desde su desarrollo en la década de 1970, FTP ha sido una herramienta fundamental

para la transferencia de datos, especialmente en entornos donde se requiere el intercambio eficiente y confiable de archivos.

El funcionamiento básico de FTP implica la conexión de un cliente FTP a un servidor FTP a través de una conexión de red. Una vez establecida la conexión, el cliente puede enviar comandos al servidor para realizar varias operaciones, como listar los archivos en el servidor, descargar archivos del servidor al cliente, cargar archivos desde el cliente al servidor y eliminar archivos en el servidor.

Algunas características importantes del Protocolo FTP incluyen:

- **Modos de Transferencia**: FTP admite dos modos de transferencia de datos: modo de transferencia activo y modo de transferencia pasivo. En el modo activo, el servidor inicia la conexión de datos con el cliente, mientras que en el modo pasivo, el cliente inicia la conexión de datos con el servidor. El modo pasivo es comúnmente utilizado en entornos donde el cliente se encuentra detrás de un firewall o un servidor proxy.

- **Autenticación**: FTP proporciona mecanismos de autenticación para garantizar la seguridad durante la transferencia de archivos. Los usuarios pueden autenticarse en el servidor FTP utilizando diferentes métodos, como la autenticación basada en contraseñas, la autenticación basada en claves públicas o la autenticación anónima.

- **Seguridad**: Aunque FTP es ampliamente utilizado, es importante destacar que no proporciona cifrado de datos de forma predeterminada, lo que significa que los datos transferidos a través de FTP pueden ser interceptados y leídos por terceros. Para mitigar este riesgo, se han desarrollado extensiones como FTPS (FTP seguro) y SFTP (FTP seguro por SSH), que proporcionan capas adicionales de seguridad mediante el cifrado de datos durante la transferencia.

- **Flexibilidad y Versatilidad**: FTP es una herramienta altamente flexible y versátil que se puede utilizar en una variedad de entornos y aplicaciones. Desde la transferencia de archivos simples entre usuarios individuales hasta la distribución de grandes volúmenes de datos entre servidores en entornos empresariales, FTP ofrece una solución robusta y confiable para la transferencia de archivos en la red.

El **Protocolo SSH (Secure Shell)** es un protocolo de red que proporciona una forma segura de acceder y administrar servidores remotos a través de una conexión cifrada. SSH fue diseñado como una alternativa segura a otros protocolos de acceso remoto, como Telnet, que transmiten datos en texto sin cifrar, lo que los hace vulnerables a la interceptación y manipulación por parte de terceros.

Algunas características clave del Protocolo SSH incluyen:

- **Cifrado de Datos**: SSH utiliza algoritmos de cifrado para proteger la integridad y confidencialidad de los datos durante la transmisión entre el cliente y el servidor. Estos algoritmos cifran los datos antes de ser enviados y los descifran una vez recibidos, garantizando que solo el cliente y el servidor autorizados puedan acceder a la información.

- **Autenticación del Servidor y del Cliente**: SSH permite la autenticación tanto del servidor como del cliente, lo que garantiza que ambas partes sean quienes afirman ser. La autenticación del servidor se realiza mediante el intercambio de claves públicas y privadas entre el cliente y el servidor, mientras que la autenticación del cliente puede basarse en contraseñas, claves públicas, tokens de seguridad o cualquier otro mecanismo de autenticación admitido.

- **Túneles Seguros**: SSH puede crear túneles seguros a través de una red no segura, permitiendo que el tráfico de datos se transmita de manera segura entre el cliente y el servidor. Esto es especialmente útil para acceder a servicios y recursos internos de una red privada desde ubicaciones externas, como el acceso a bases de datos, sistemas de archivos compartidos o aplicaciones web.

- **Gestión de Sesiones Interactivas y No Interactivas**: SSH es compatible con la ejecución de comandos remotos de manera interactiva, lo que permite a los usuarios administrar servidores y sistemas remotos a través de una interfaz de línea de comandos. También es posible ejecutar comandos de manera no interactiva, lo que facilita la automatización de tareas administrativas y la ejecución de scripts en servidores remotos.

¿Cómo funciona SSH?

Si usas Linux o Mac, entonces usar el protocolo SSH es muy fácil. Si utilizas Windows, deberás utilizar un cliente SSH para abrir conexiones SSH. El cliente SSH más popular es PuTTY, puedes aprender más acerca de él aquí.

Para usuarios de Mac y Linux, dirígete a tu programa de terminal y sigue este procedimiento:

El comando SSH consta de 3 partes distintas:

ssh {user}@{host}

El comando de clave SSH le indica a tu sistema que desea abrir una Conexión de Shell Segura y cifrada.

- **{user}** representa la cuenta a la que deseas acceder. Por ejemplo, puede que quieras acceder al usuario root, que es básicamente para el administrador del sistema con derechos completos para modificar cualquier cosa en el sistema.

- **{host}** hace referencia al equipo al que quieres acceder. Esto puede ser una dirección IP (por ejemplo, 244.235.23.19) o un nombre de dominio (por ejemplo, www.xyzdomain.com).

Al pulsar enter, se te pedirá que escribas la contraseña de la cuenta solicitada. Al escribirla, nada aparecerá en la pantalla, pero tu contraseña, de hecho, se está transmitiendo. Una vez que hayas terminado de escribir, pulsa enter una vez más. Si tu contraseña es correcta, verás una ventana de terminal remota.

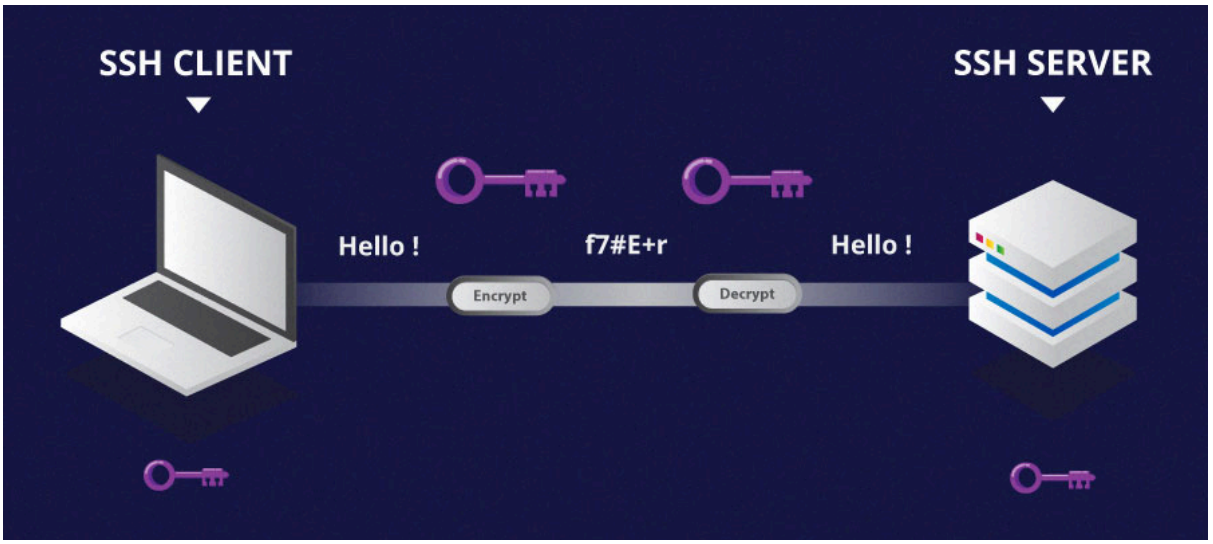
Técnicas de cifrado

La ventaja significativa ofrecida por el protocolo SSH sobre sus predecesores es el uso del cifrado para asegurar la transferencia segura de información entre el host y el cliente. Host se refiere al servidor remoto al que estás intentando acceder, mientras que el cliente es el equipo que estás utilizando para acceder al host. Hay tres tecnologías de cifrado diferentes utilizadas por SSH:

1. Cifrado simétrico
2. Cifrado asimétrico
3. Hashing

Cifrado Simétrico

El cifrado simétrico es una forma de cifrado en la que se utiliza una clave secreta tanto para el cifrado como para el descifrado de un mensaje, tanto por el cliente como por el host. Efectivamente, cualquiera que tenga la clave puede descifrar el mensaje que se transfiere.



El cifrado simétrico a menudo se llama clave compartida (shared key) o cifrado secreto compartido. Normalmente sólo hay una clave que se utiliza, o a veces un par de claves donde una clave se puede calcular fácilmente con la otra clave.

Las claves simétricas se utilizan para cifrar toda la comunicación durante una sesión SSH. Tanto el cliente como el servidor derivan la clave secreta utilizando un método acordado, y la clave resultante nunca se revela a terceros. El proceso de creación de una clave simétrica se lleva a cabo mediante un algoritmo de intercambio de claves.

Lo que hace que este algoritmo sea particularmente seguro es el hecho de que la clave nunca se transmite entre el cliente y el host. En lugar de eso, los dos equipos comparten datos públicos y luego los manipulan para calcular de forma independiente la clave secreta. Incluso si otra máquina captura los datos públicamente compartidos, no será capaz de calcular la clave porque el algoritmo de intercambio de clave no se conoce.

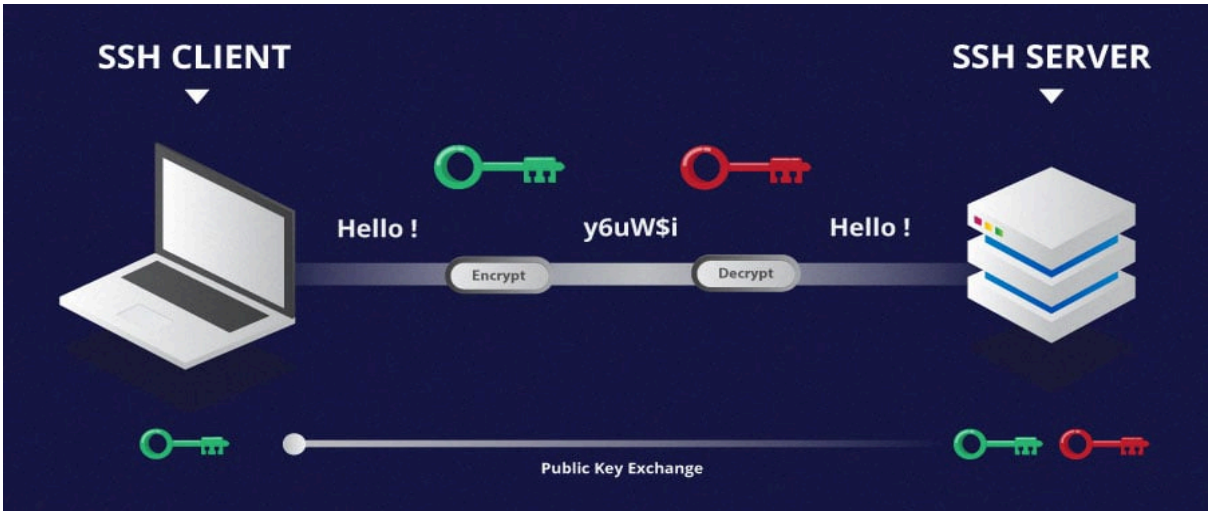
Debe tenerse en cuenta, sin embargo, que el token secreto es específico para cada sesión SSH, y se genera antes de la autenticación del cliente. Una vez generada la clave, todos los paquetes que se mueven entre las dos máquinas deben ser cifrados por la clave privada. Esto incluye la contraseña escrita en la consola por el usuario, por lo que las credenciales siempre están protegidas de los fisgones de paquetes de red.

Existen varios códigos cifrados simétricos, incluyendo, pero no limitado a, AES (Advanced Encryption Standard), CAST128, Blowfish, etc. Antes de establecer una conexión segura, el cliente y un host deciden qué cifrado usar, publicando una lista de cifrados soportados por orden de preferencia. El cifrado preferido de entre los soportados por los clientes que está presente en la lista del host se utiliza como el cifrado bidireccional.

Por ejemplo, si dos máquinas Ubuntu 14.04 LTS se comunican entre sí a través de SSH, utilizarán aes128-ctr como su cifrado predeterminado.

Cifrado Asimétrico

A diferencia del cifrado simétrico, el cifrado asimétrico utiliza dos claves separadas para el cifrado y el descifrado. Estas dos claves se conocen como la clave pública (public key) y la clave privada (private key). Juntas, estas claves forman el par de claves pública-privada (public-private key pair).



La clave pública, como sugiere el nombre, se distribuye abiertamente y se comparte con todas las partes. Si bien está estrechamente vinculado con la clave privada en términos de funcionalidad, la clave privada no se puede calcular matemáticamente desde la clave pública. La relación entre las dos claves es altamente compleja: un mensaje cifrado por la clave pública de una máquina, sólo puede ser descifrado por la misma clave privada de la máquina. Esta relación unidireccional significa que la clave pública no puede descifrar sus propios mensajes ni descifrar nada cifrado por la clave privada.

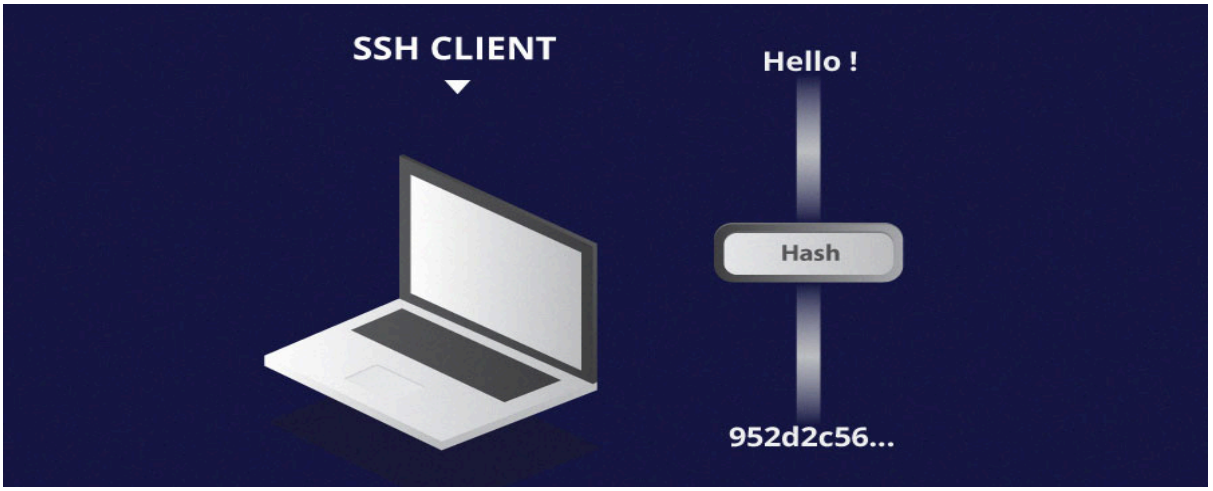
La clave privada debe permanecer privada, es decir, para que la conexión sea asegura, ningún tercero debe conocerla. La fuerza de toda la conexión reside en el hecho de que la clave privada nunca se revela, ya que es el único componente capaz de descifrar mensajes que fueron cifrados usando su propia clave pública. Por lo tanto, cualquier parte con la capacidad de descifrar mensajes firmados públicamente debe poseer la clave privada correspondiente.

A diferencia de la percepción general, el cifrado asimétrico no se utiliza para cifrar toda la sesión SSH. En lugar de eso, sólo se utiliza durante el algoritmo de intercambio de claves de cifrado simétrico. Antes de iniciar una conexión segura, ambas partes generan pares de claves públicas-privadas temporales y comparten sus respectivas claves privadas para producir la clave secreta compartida.

Una vez que se ha establecido una comunicación simétrica segura, el servidor utiliza la clave pública de los clientes para generar y desafiar y transmitirla al cliente para su autenticación. Si el cliente puede descifrar correctamente el mensaje, significa que contiene la clave privada necesaria para la conexión. Y entonces comienza la sesión SSH.

Hashing

El hashing unidireccional es otra forma de criptografía utilizada en Secure Shell Connections. Las funciones de hash unidireccionales difieren de las dos formas anteriores de encriptación en el sentido de que nunca están destinadas a ser descifradas. Generan un valor único de una longitud fija para cada entrada que no muestra una tendencia clara que pueda explotarse. Esto los hace prácticamente imposibles de revertir.



Es fácil generar un hash criptográfico de una entrada dada, pero imposible de generar la entrada del hash. Esto significa que si un cliente tiene la entrada correcta, pueden generar el hash criptográfico y comparar su valor para verificar si poseen la entrada correcta.

SSH utiliza hashes para verificar la autenticidad de los mensajes. Esto se hace usando HMACs, o códigos de autenticación de mensajes basados en hash. Esto asegura que el comando recibido no se altere de ninguna manera.

Puedes probar generar tus propios hash desde https://www.freeformatter.com/hmac-generator.html.

Mientras se selecciona el algoritmo de cifrado simétrico, también se selecciona un algoritmo de autenticación de mensajes adecuado. Esto funciona de manera similar a cómo se selecciona el cifrado, como se explica en la sección de cifrado simétrico.

Todo mensaje transmitido debe contener un MAC, que se calcula utilizando la clave simétrica, el número de secuencia de paquetes y el contenido del mensaje. Se envía fuera de los datos cifrados simétricamente como la sección final del paquete de comunicaciones.

Negociación de cifrado de sesión

Cuando un cliente intenta conectarse al servidor a través de TCP, el servidor presenta los protocolos de cifrado y las versiones respectivas que soporta. Si el cliente tiene un par similar de protocolo y versión, se alcanza un acuerdo y se inicia la conexión con el protocolo aceptado. El servidor también utiliza una clave pública asimétrica que el cliente puede utilizar para verificar la autenticidad del host.

Una vez que esto se establece, las dos partes usan lo que se conoce como Algoritmo de Intercambio de Claves Diffie-Hellman para crear una clave simétrica. Este algoritmo permite que tanto el cliente como el servidor lleguen a una clave de cifrado compartida que se utilizará en adelante para cifrar toda la sesión de comunicación.

Aquí es cómo el algoritmo trabaja en un nivel muy básico:

1. Tanto el cliente como el servidor coinciden en un número primo muy grande, que por supuesto no tiene ningún factor en común. Este valor de número primo también se conoce como el valor semilla **(seed value)**.

2. Luego, las dos partes acuerdan un mecanismo de cifrado común para generar otro conjunto de valores manipulando los valores semilla de una manera algorítmica específica. Estos mecanismos, también conocidos como generadores de cifrado, realizan grandes operaciones sobre la semilla. Un ejemplo de dicho generador es AES (Advanced Encryption Standard).

3. Ambas partes generan independientemente otro número primo. Esto se utiliza como una clave **privada secreta para la interacción**.

4. Esta clave privada recién generada, con el número compartido y el algoritmo de cifrado (por ejemplo, AES), se utiliza para calcular una **clave pública** que se distribuye a la otra computadora.

5. A continuación, las partes utilizan su clave privada personal, la clave pública compartida de la otra máquina y el número primo original para crear una clave compartida final. Esta clave se calcula de forma independiente por ambos equipos, pero creará la misma clave de cifrado en ambos lados.

6. Ahora que ambas partes tienen una clave compartida, pueden cifrar simétricamente toda la sesión SSH. La misma clave se puede utilizar para cifrar y descifrar mensajes (leer: sección sobre cifrado simétrico).

Ahora que se ha establecido la sesión cifrada segura simétricamente, el usuario debe ser autenticado.

Uniform Resource Locators - URL

Una **URL (Uniform Resource Locator)** es una dirección que se utiliza para identificar de manera única un recurso en internet. Básicamente, una URL indica la ubicación de un recurso específico, como una página web, un archivo, una imagen o un servicio, en la web.

- **Protocolo**: El protocolo es la parte inicial de una URL y define cómo se debe acceder al recurso. Los protocolos más comunes son HTTP (Hypertext Transfer Protocol) y HTTPS (HTTP Secure) para páginas web, FTP (File Transfer Protocol) para transferencia de archivos, y mailto para direcciones de correo electrónico.

- **Dominio**: El dominio, también conocido como nombre de host, identifica la ubicación específica en la web donde se encuentra el recurso. Por ejemplo, en la URL "https://www.ejemplo.com", "www.ejemplo.com" es el dominio.

- **Ruta**: La ruta especifica la ubicación exacta del recurso dentro del servidor. Por ejemplo, en la URL "https://www.ejemplo.com/pagina.html", "/pagina.html" es la ruta que indica el nombre y la ubicación del archivo en el servidor.

- **Parámetros**: Los parámetros son datos adicionales que se pueden incluir en la URL para proporcionar información adicional al servidor. Por ejemplo, en la URL "https://www.ejemplo.com/busqueda?q=ejemplo", "?q=ejemplo" es un parámetro que indica que se está realizando una búsqueda con la palabra "ejemplo".

- **Fragmento**: El fragmento, también conocido como ancla, identifica una sección específica dentro de un recurso más grande, como una página web. Por ejemplo, en la URL "https://www.ejemplo.com/pagina.html#seccion", "#seccion" indica que se debe mostrar la sección específica llamada "seccion" en la página web.

Ejemplo:

https://www.ejemplo.com/ruta/pagina.html?parametro1=valor1&parametro2=valor2#seccion

Desglose de los componentes:

- Protocolo: https://
- Dominio: www.ejemplo.com
- Ruta: /ruta/pagina.html
- Parámetros: ?parametro1=valor1&parametro2=valor2
- Fragmento: #seccion

Las URL solo se pueden enviar a través de Internet utilizando el juego de caracteres ASCII. **Si una URL contiene caracteres fuera del conjunto ASCII, la URL debe convertirse**.

La codificación de URL convierte caracteres que no son ASCII a un formato que se puede transmitir a través de Internet.

La codificación de URL reemplaza los caracteres que no son ASCII con un **"%"** seguido de dígitos hexadecimales.

Las URL no pueden contener espacios. La codificación de URL normalmente reemplaza un espacio con un signo más (+), o **%20**.

Ejemplos de codificación ASCII

Su navegador codificará la entrada, de acuerdo con el conjunto de caracteres utilizado en su página.

El juego de caracteres predeterminado en HTML5 es UTF-8.

Personaje	Desde Windows-1252	Desde UTF-8
€	%80	%E2%82%AC
£	%A3	%C2%A3
©	%A9	%C2%A9
®	%AE	%C2%AE
À	%C0	%C3%80
Á	%C1	%C3%81
Â	%C2	%C3%82
Ã	%C3	%C3%83
Ä	%C4	%C3%84
Å	%C5	%C3%85

Queridos alumnos,

Les animamos a ir más allá de los materiales que les damos en clase. Exploren la biblioteca de la facultad, busquen en internet y descubran nuevos recursos que les ayuden a entender mejor los temas. Investigar por su cuenta no solo les dará una visión más amplia, sino que también hará el aprendizaje mucho más emocionante.