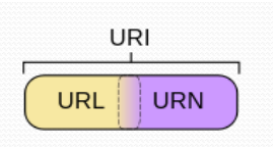
TEMA 7 – Web Servers y seguridad

Objetivos

1. Entender las arquitecturas de Web Services
2. Identificar los componentes más importantes de Secure Services
3. Introducir certificados y public keys

Definiciones importantes

- **Servicio** → Mecanismo que permite el acceso a una o más competencias (desarrollar algo), donde el acceso se proporciona mediante una interfaz prescrita y se ejerce con las restricciones y políticas especificadas en la descripción del servicio.
- **Servicio Orientado a Arquitecturas o objetos** (SOA) → Paradigma para organizar y utilizar competencias distribuidas que pueden estar bajo el control de diferentes dominios de propiedad.
- **Sistema distribuido** → Consiste en diversos agentes de software (programa que actúa para un usuario u otro programa) que deben trabajar juntos para realizar algunas tareas. NO operan en el mismo entorno de procesamiento, por tanto, se tienen que comunicar con algún protocolo. SOA es un sistema distribuido.
- **URI** → Es una cadena de caracteres que se utiliza para identificar un nombre de un recurso web. Un recurso puede ser cualquier cosa que tenga una identidad. Un URI es un URL, un URN (URI que no especifica la ubicación, es un espacio de nombres) o ambos. (como en la imagen)



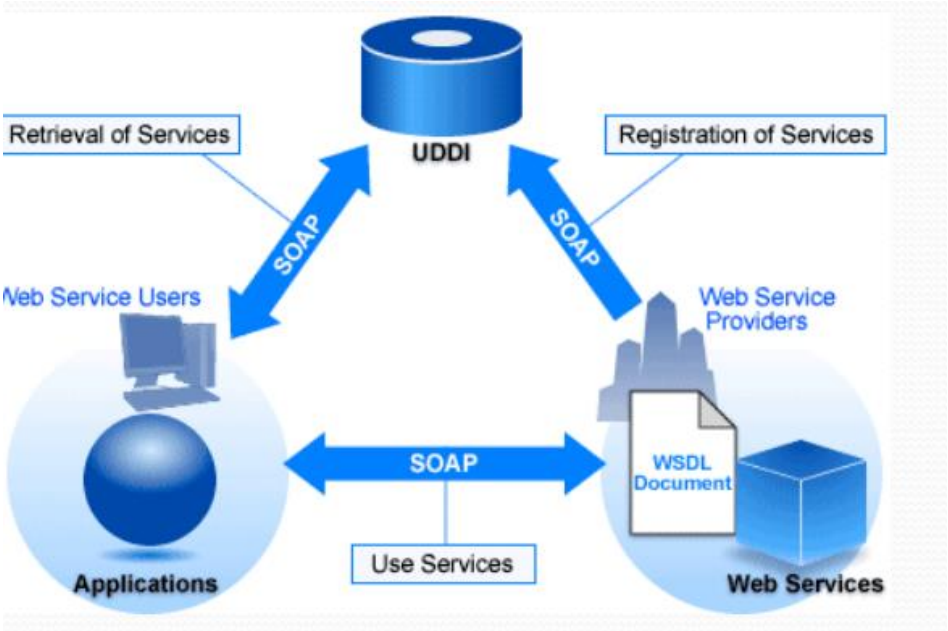
Web service

- **Proporcionan un medio estándar de interoperación entre diferentes aplicaciones de software**, que se ejecutan en una variedad de plataformas.
- Ejemplo de web service:

  Construyes una página web que incluye el tiempo de Barcelona. El instituto Catalán de Previsión tiene una aplicación que ofrece un mapa del tiempo actual en cualquier ciudad de Cataluña. Para nuestra página web, tendremos que ir solicitando el mapa del tiempo al Instituto Catalán de Previsión (solicitamos una información)
- Tenemos dos tipos de web services, REST (manipulando representaciones XML/JSON de los recursos web y típicamente usa HTTP para la transferencia) o servicios web arbitrarios.
- Un Web Service es un sistema de software diseñado para soportar interacción de máquina a máquina a través de una red.
- **Tiene una interfaz descrita en un formato procesable por máquina**. → **WSDL** (Web Services Description Language). Se usa a menudo en combinación con SOAP.

  Lenguaje **basado en XML** que proporciona un modelo para **describir los servicios que ofrece y su localización y los parámetros y métodos que admiten.**

- **Otros sistemas interactúan con el servicio web de una manera prescrita** por su descripción **mediante el uso de mensajes SOAP,** generalmente transmitidos mediante HTTP.
- **Estructura XML Web Service:** → Es un sistema de software identificado por un URI, cuyas interfaces y enlaces públicos se definen y describen mediante XML.

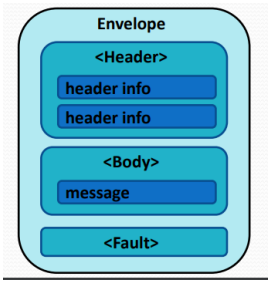


Requester → Aplicación que pide la ejecución de un Web Service. (Web Service User)

Provider → Procesa la petición, sitio donde el requester accede para el servicio. Es el propietario del servicio.

Discovery agency → (OPCIONAL) Sitio a través de la cual se publica una descripción del servicio web por parte del provider y se hace visible para los requesters. (UDDI)

Su objetivo es ser accedido por mensajes SOAP y dar paso a documentos WSDL los cuales describen los requisitos del protocolo y formatos del mensaje.

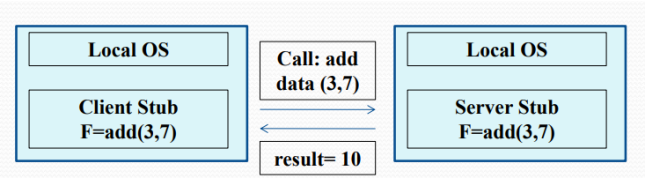


+ SOAP (Simple Object Access Protocol) → Protocolo para el intercambio de información estructurada entre el cliente y los servidores en servicios web. Usa formato XML para los mensajes y HTTP o SMTP. Estructura (no sale en examen)

(ejemplo uso pag 18)

- **RPC** (Remote Call Procedure)

Permite la invocación de una función específica (código) de otra máquina a través de una red. Es inicializado por el cliente que envía un mensaje de solicitud a un servidor remoto conocido. El servidor remoto envía la respuesta al cliente y la aplicación continua su proceso con el resultado que le ha proporcionado. (se pasan parámetros también)



- **XML-RPC**

  Es una versión de RPC que usa XML para serializar parámetros. Usa HTTP como protocolo de transporte y define pocos formatos de datos. (ejemplo de uso pag 17)

- **JSON Web Services** (diferente de las basadas en XML)

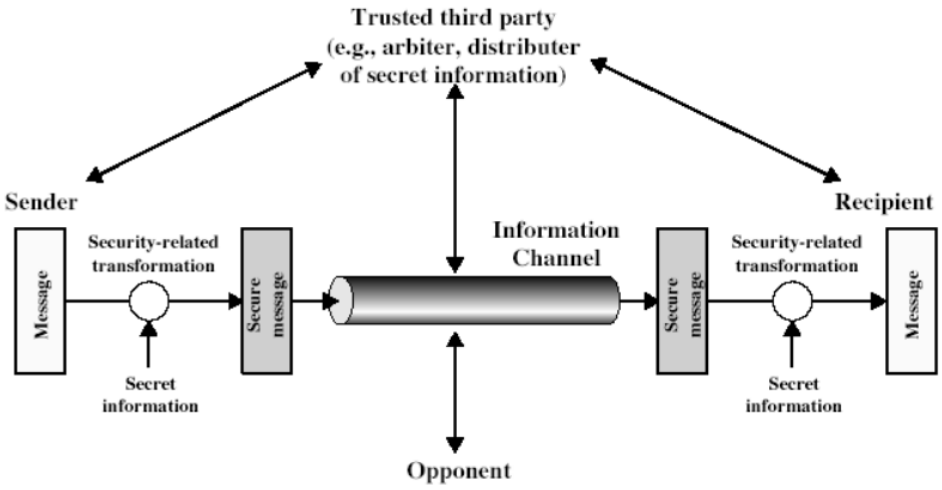
  Es una alternativa a XML. Se usa a menudo para serializar (proceso de convertir un objeto en una secuencia de bytes para almacenar) y transmitir datos estructurados a través de una conexión de red.

  A diferencia de XML, permite **notificaciones** (información enviada al servidor que no requiere una respuesta) y envío de varias llamadas al servidor para ser contestadas fuera de un orden. Y, además, es menos complejo que XML.

<u>Seguridad</u>

- Definiciones:
  - **Computer Security** → Colección de **herramientas diseñadas para proteger datos y para evitar hackers**.
  - **Network Security** → Medidas para **proteger los datos durante su transmisión**
  - **Internet Security** → Medidas para proteger los datos durante su transmisión a **través de una colección de redes interconectadas.**

- **Modelo de seguridad en una red**:



  Se necesita para ello, **un algoritmo adecuado** para la security transformation, generar información secreta (**uso de keys**) y desarrollar **métodos para distribuir y compartir** la **información secreta**.

- **Criptografía de clave**

  Se usan claves para cifrar/descifrar los mensajes

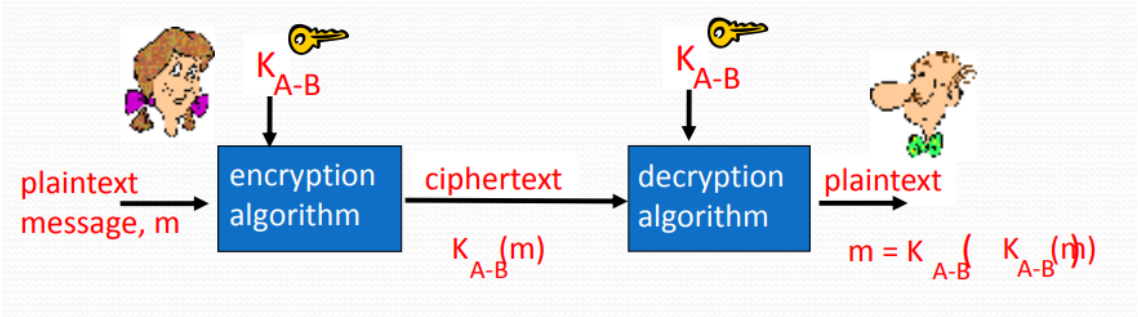
  E.g.: **Plaintext (m): bob. i love you. alice**
  Ciphertext (c): nkn. s gktc wky. mgsbc

  c = K(m)

  + **cifrado de sustitución** → Sustituir una cosa por otro (en el ejemplo, por otra letra)

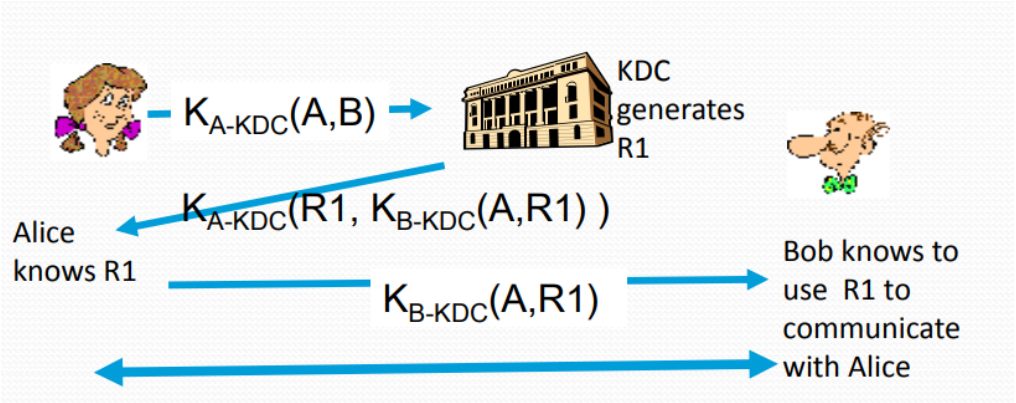
+ Son necesarias dos claves, una para encriptar y otra para desencriptar. La criptografía tradicional de clave privada / secreta / única usa una clave.

- **Criptografía de clave simétrica**: (funcionamiento)



Bob y Alice comparten la misma clave simétrica conocida: Ka-b (cifrado mono alfabético).

¿Cómo concuerdan Bob y Alice el valor de la clave? → Confían en **KDC** (Key Distribution Center) que actúa como intermediario.



Ahora, Alice y Bob se comunican usando R1 como **clave de sesión** para el cifrado simétrico compartido

- **Criptografía de clave pública o asimétrica**

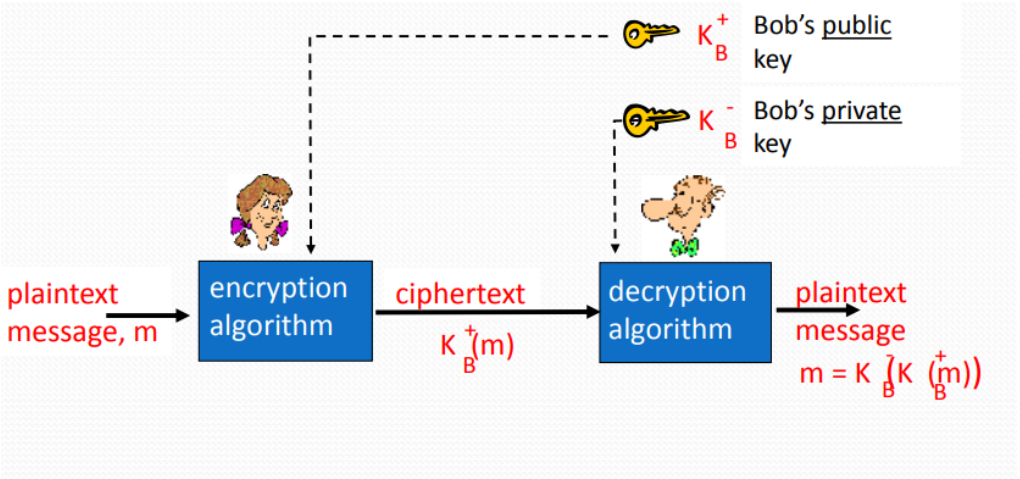
Es **asimétrica** porque **aquellos que cifran mensajes o verifican firmas no pueden descifrar mensajes ni crear.**

Utiliza dos claves:

- **Clave pública** → Que puede ser conocida por cualquier persona, y se puede usar para cifrar mensajes y verificar firmas.
- **Clave privada** → Conocida solo por el destinatario, se utiliza para descifrar mensajes y crear firmas

Se requiere que existan Kb+ (pública) y Kb-(privada) y que dado una clave pública Kb+ sea imposible calcular una privada Kb- a partir de la pública. El algoritmo que hace esto se llama **RSA**.

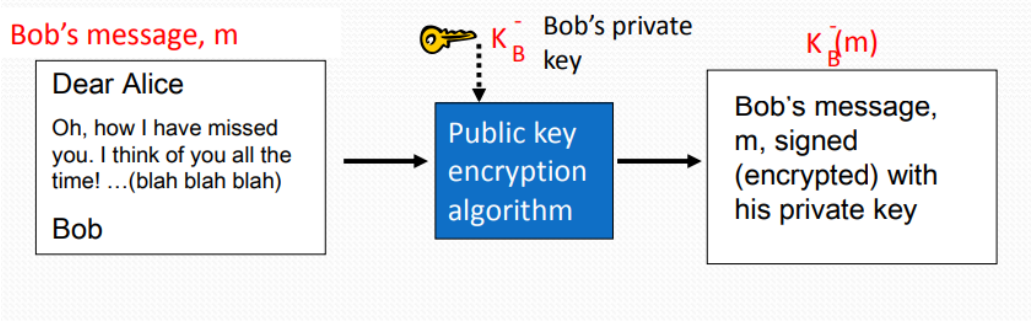
Funcionamiento:



¿Por qué la criptografía de clave pública?

Para dos cosas principalmente; **Distribución de claves** (tener comunicaciones seguras sin tener que confiar en KDC) y **Firmas digitales** (para verificar que el mensaje viene intacto)

Funcionamiento firma digital: (con ejemplo)



1. Bob signa m encriptando con su clave privada Kb-, creando Kb-(m)

2. Alice recibe el mensaje m con la firma digital Kb-(m)

3. Alice verifica m (firmado por Bob) aplicando la clave pública de Bob con Kb-(m). Entonces comprueba lo siguiente: $K_B^+(K_B^-(m)) = m$,

4.Si se cumple la condición anterior, Alice verifica que Bob firmó m y que nadie más lo firmó.

- **Certification authority** (CA)

Une la clave pública a una entidad particular, llamémosle E. Si eso pasa, E proporciona "prueba de identidad" a CA y CA crea el certificado de enlace E a su clave pública. Este certificado contiene la clave pública de E firmada digitalmente por CA.