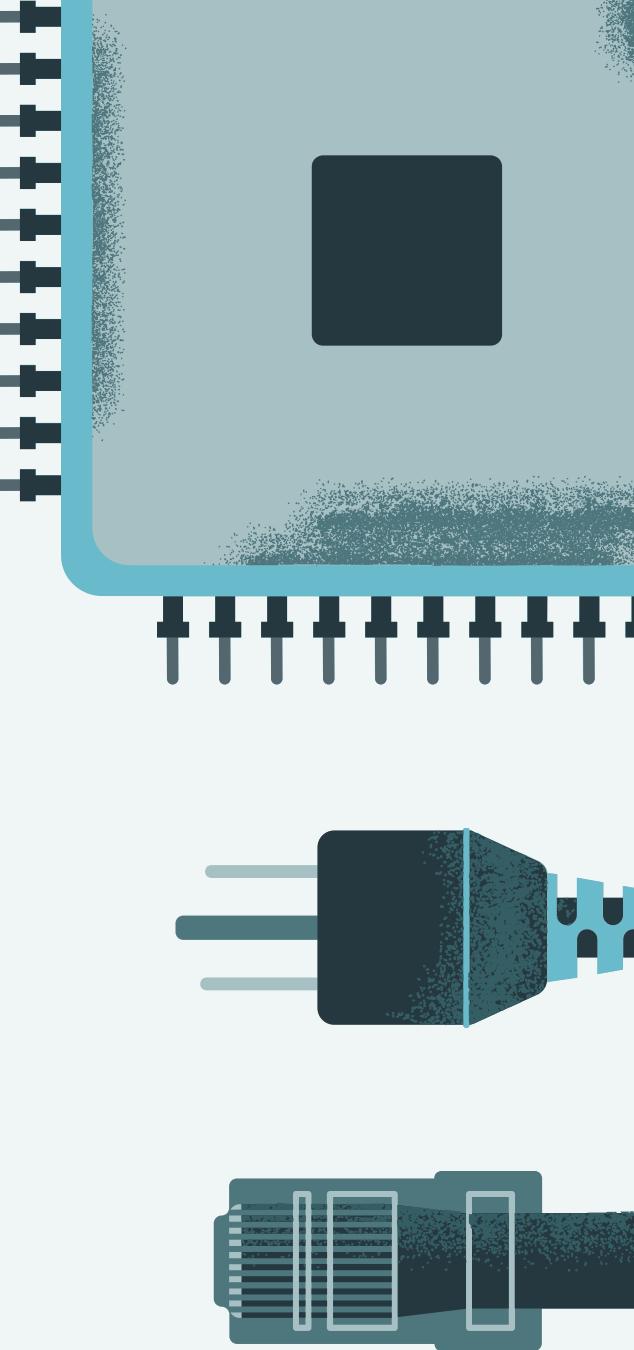
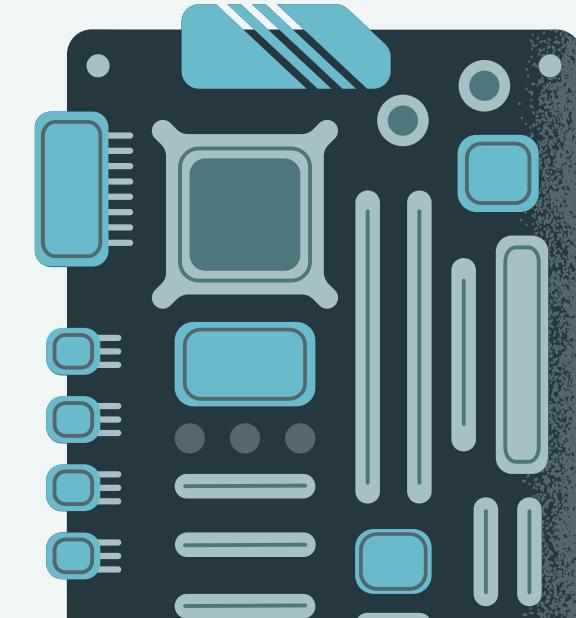
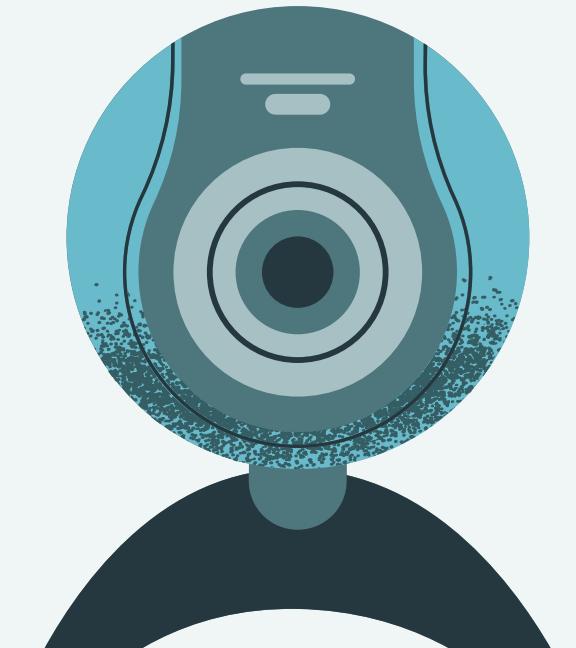
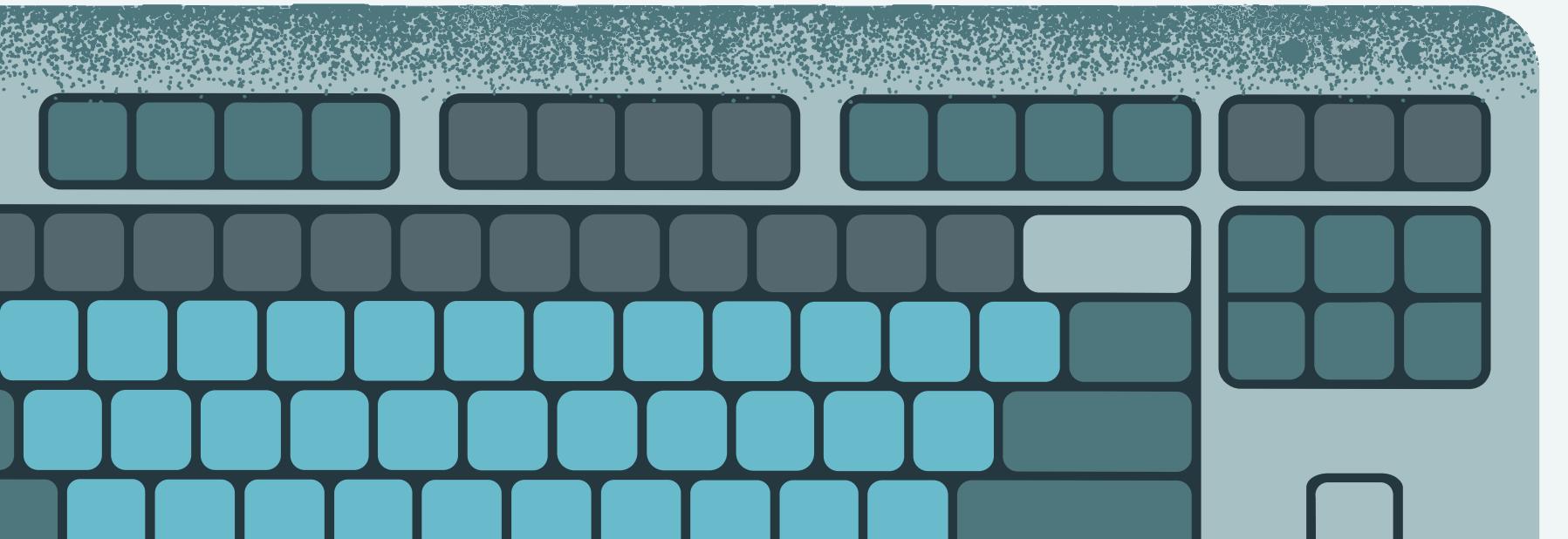


SERVIDORES HTTPS



INTRODUCCIÓN

¿QUÉ ES HTTPS Y POR QUÉ ES IMPORTANTE?

HTTPS (HYPERTEXT TRANSFER PROTOCOL SECURE) ES UNA VERSIÓN SEGURA DEL PROTOCOLO HTTP. UTILIZA TLS (TRANSPORT LAYER SECURITY) O SU PREDECESOR SSL (SECURE SOCKETS LAYER) PARA CIFRAR LA COMUNICACIÓN ENTRE UN CLIENTE (NORMALMENTE UN NAVEGADOR) Y UN SERVIDOR WEB.



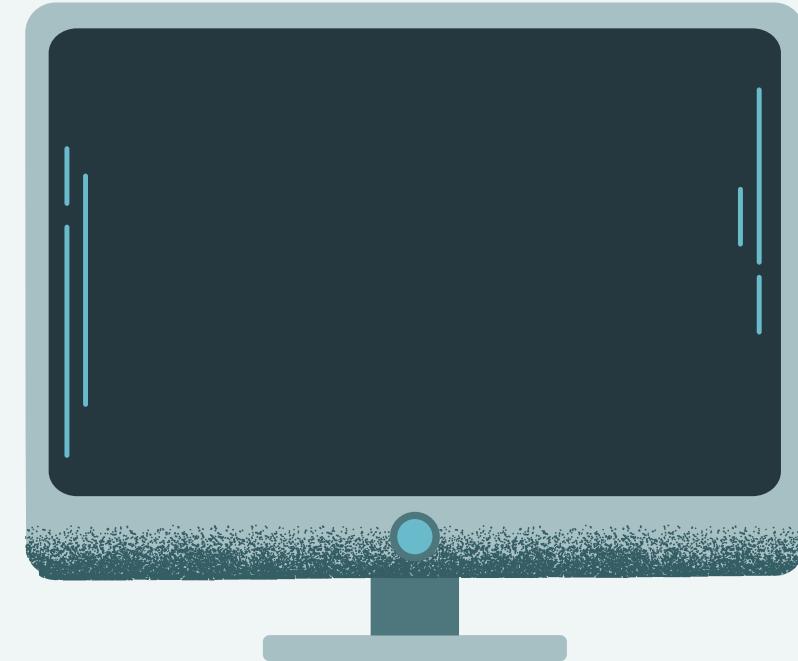
Importancia:

- **Confidencialidad:** Protege los datos sensibles del usuario (contraseñas, datos bancarios, etc.) al cifrarlos.
- **Integridad:** Asegura que los datos no sean modificados durante la transmisión.
- **Autenticación:** Permite verificar que el servidor es legítimo mediante certificados digitales.
- **Confianza** del usuario: Los navegadores modernos alertan a los usuarios si un sitio no es seguro, lo que puede afectar la reputación de un sitio web.

DIFERENCIAS ENTRE HTTP Y HTTPS

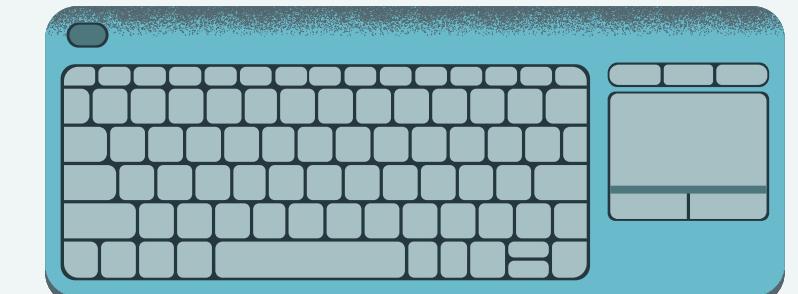
HTTP

- NO TIENE CIFRADO
- PUERTO 80
- NO REQUIERE CERTIFICADO
- PROTECCION DE DATOS INEXISTENTE
- NINGUN INDICADOR EN EL NAVEGADOR
- VULNERABLE A ATAQUES MITM (Hombre en Medio)



HTTPS

- CIFRADO MEDIANTE TLS/SSL
- PUERTO 443
- REQUIERE CERTIFICADO TLS
- DATOS CIFRADOS Y PROTEGIDOS
- CANDADO Y NOMBRE VERIFICADO (Navegador)
- PROTEGE CONTRA ATAQUES MITM (Hombre en Medio)



CONCEPTOS BÁSICOS DE SEGURIDAD

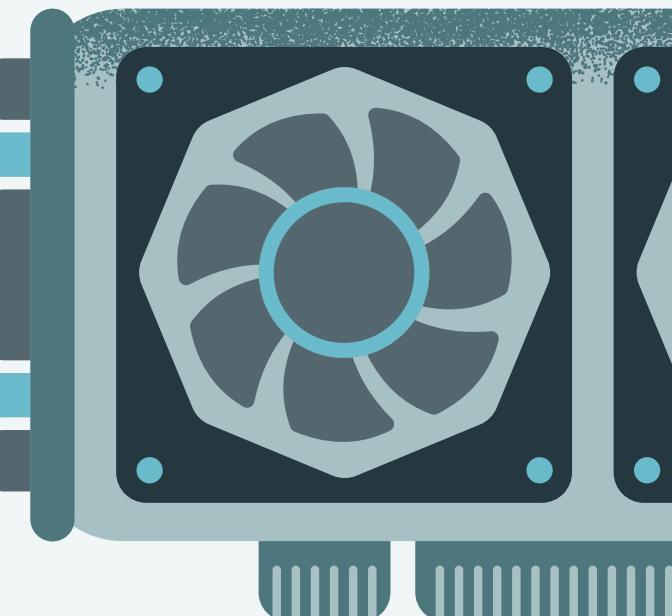


◆ TLS/SSL

TLS (Transport Layer Security) es el protocolo de seguridad estándar que reemplazó a SSL. Crea un canal cifrado entre el navegador y el servidor. Usa un protocolo de negociación (handshake) para establecer claves de cifrado de forma segura.

◆ CERTIFICADOS DIGITALES

- *Emitidos por una Autoridad de Certificación (CA) confiable.*
- *Contienen:*
 - Nombre del dominio
 - Clave pública
 - Firma digital de la CA
 - Fecha de expiración
- *Verifican que el servidor es quien dice ser.*





¿CÓMO FUNCIONA HTTPS?

HTTPS asegura la comunicación entre el cliente (navegador) y el servidor web mediante TLS (Transport Layer Security). A continuación, se explican sus componentes y el proceso de forma simplificada.

1. HANDSHAKE TLS (DE FORMA SIMPLIFICADA)

El TLS Handshake es el proceso inicial mediante el cual el cliente y el servidor acuerdan cómo comunicarse de forma segura. Ocurre en milisegundos.

Pasos simplificados:

1. **Cliente → Servidor:** El cliente inicia la conexión y envía una lista de versiones TLS y cifrados que soporta.
2. **Servidor → Cliente:**
 - Responde con el cifrado seleccionado.
 - Envía su certificado digital (incluye la clave pública).
3. **Cliente:**
 - Verifica el certificado (firma digital, validez, CA, etc.).
 - Genera una clave de sesión aleatoria.
 - La cifra con la clave pública del servidor y la envía.
4. **Servidor:**
 - Usa su clave privada para descifrar la clave de sesión.
5. Ambos ahora comparten la misma clave de sesión y comienzan la comunicación cifrada simétrica.

CERTIFICADOS DIGITALES Y AUTORIDADES CERTIFICADORAS (CA)

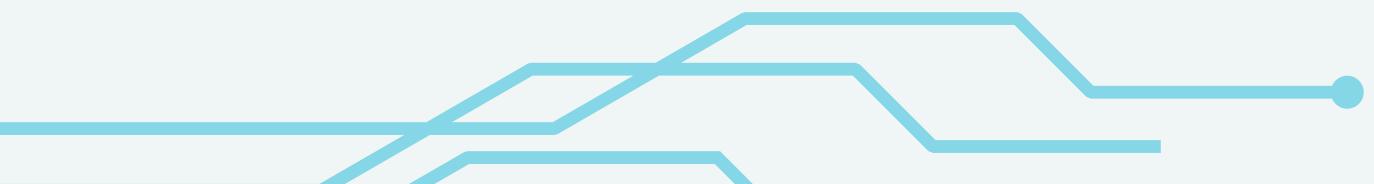
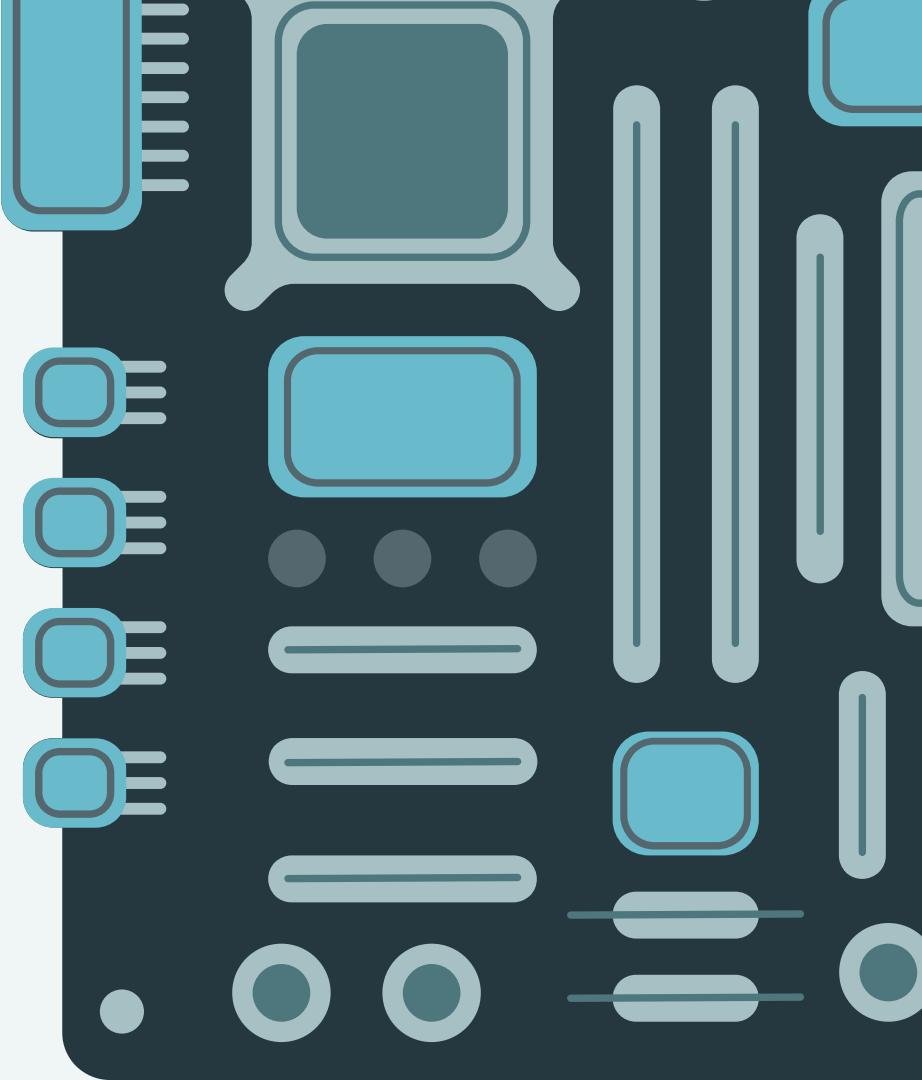
UN CERTIFICADO DIGITAL ES UN ARCHIVO QUE VALIDA LA IDENTIDAD DEL SERVIDOR. ESTÁ EMITIDO Y FIRMADO POR UNA AUTORIDAD CERTIFICADORA (CA) CONFIABLE.

CONTIENE:

- Nombre del dominio.
- Clave pública del servidor.
- Información de la CA.
- Fecha de validez.
- Firma digital de la CA.

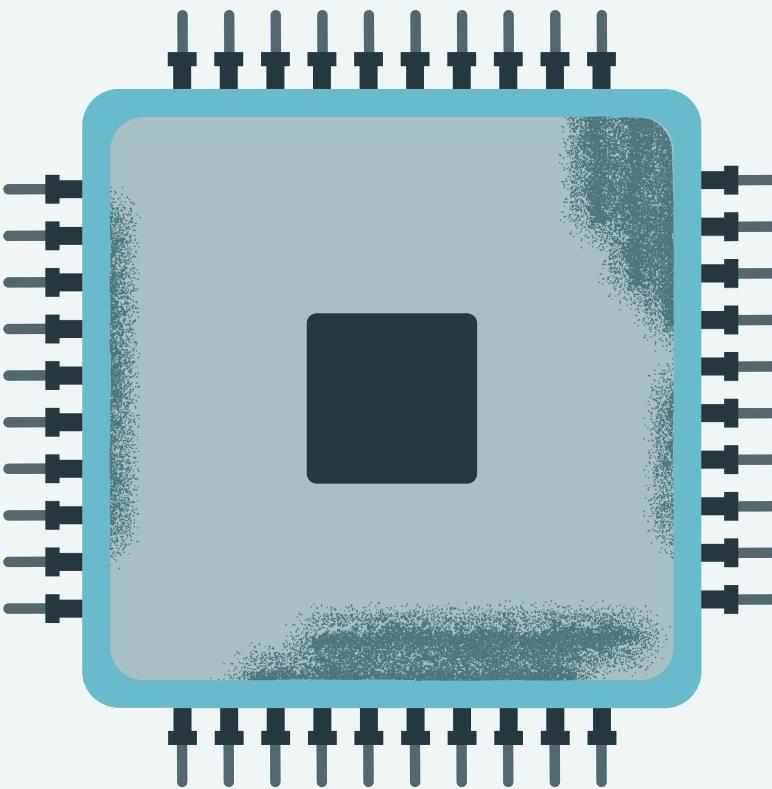
¿QUÉ HACE LA CA?

- Verifica que el propietario del certificado realmente controla el dominio.
 - Firma digitalmente el certificado para que los navegadores puedan confiar en él.
- Los navegadores y sistemas operativos tienen una lista de CA raíz confiables preinstalada.



CLAVES PÚBLICAS Y PRIVADAS

HTTPS USA CRIPTOGRAFÍA ASIMÉTRICA Y SIMÉTRICA:



ASIMETRICA (DURANTE EL HANDSHAKE)

- El servidor tiene:
 - Una clave pública (compartida con todos).
 - Una clave privada (secreta).
- El cliente cifra la clave de sesión con la clave pública del servidor.
- Solo el servidor puede descifrarla con su clave privada.

SIMÉTRICA (DURANTE LA COMUNICACIÓN)

- Una vez que ambas partes comparten una clave de sesión, se usa cifrado simétrico, que es más rápido.
- Ambos usan la misma clave para cifrar y descifrar los mensajes.

CONFIGURAR UN SERVIDOR HTTPS CON NODE.JS Y EXPRESS

GENERAR CERTIFICADOS AUTOFIRMADOS (PARA DESARROLLO)

En entornos de desarrollo, no necesitas una Autoridad Certificadora (CA) oficial. Puedes generar un certificado autofirmado para habilitar HTTPS localmente

```
openssl req -nodes -new -x509 -keyout key.pem -out cert.pem
```

| | |
|---------|--|
| req | Inicia una solicitud de certificado (Certificate Signing Request - CSR). |
| -nodes | Evita que se encripte la clave privada con una passphrase (útil para dev). |
| -news | Crea una nueva solicitud y clave. |
| -x509 | Genera un certificado autofirmado directamente (no una CSR para una CA). |
| -keyout | Archivo de salida para la clave privada (ej: key.pem). |
| -out | Archivo de salida para el certificado público (ej: cert.pem). |

BUENA PRACTICA

Aunque es solo para desarrollo, es buena práctica mantener los archivos seguros y organizados.

RECOMENDACIONES:

- No subir al repositorios. agregar key.pem y cert.pem a .gitignore.
- Organiza en carpetas protegidas. Por ejemplo:

```
/my-project/
└─ ssl/
    ├ key.pem
    └ cert.pem
```