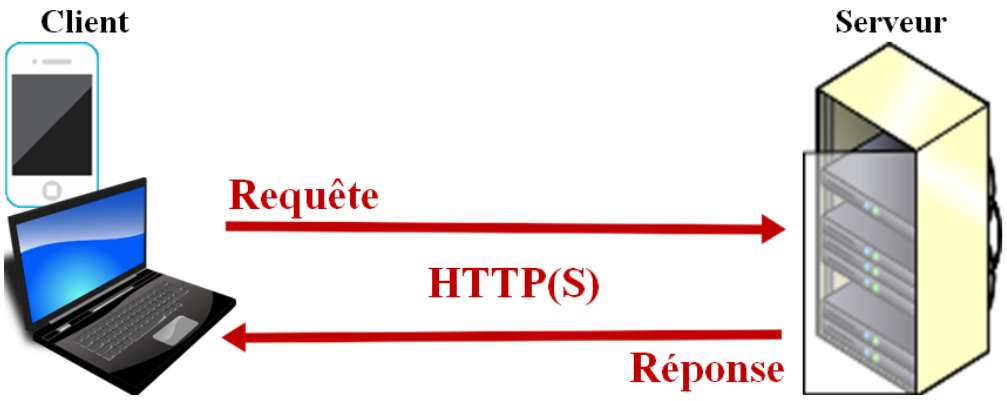
Protocole HTTP / HTTPS

HTTP : HyperText Transfer Protocol

HTTP est le protocole utilisé pour échanger des pages Web entre un navigateur et un serveur.

Le navigateur envoie une requête HTTP au serveur qui renvoie une réponse contenant la page web.



Les types de message utilisés par http sont :

- **GET** : une requête du client pour obtenir des données.
- **POST** : télécharge des fichiers de données vers le serveur web, comme des données de formulaires.
- **PUT** : télécharge des ressources ou du contenu (une image) vers le serveur.
- **DELETE** : supprime une ressource sur le serveur.
- **HEAD** : permet d'obtenir uniquement l'en-tête.

Les codes de réponse HHTP

Exemple de réponse dans le cas d'une communication client / serveur : " HTTP/1.1 200 OK … "

Type message	Code	Signification
Information	100 - Continued	La requête est en cours de traitement.
Succés	200 - OK	Requête réussie.
	201 - Created	Ressource créée avec succès.
Redirection	301 - Moved Permanently	Redirection temporaire.
	302 - Found	Redirection temporaire.
Erreur client	400 - Bad Request	Requête mal formulée.
	401 - Unauthorized	Authentification requise.
	403 - Forbidden	Accès interdit.
	404 - Not Found	Ressource introuvable.
Erreur serveur	500 - Internal Server Error	Erreur interne du serveur.
	503 - Service Unavailable	Service temporairement indisponible.

Le principal inconvénient de HTTP

HTTP n'est pas sécurisé, il transmet tout en clair. => solution HTTPS

HTTPS = HTTP + TLS

Comparatif HTTP vs HTTPS

Critère	HTTP	HTTPS
Port	80	443
Chiffrement	Non	Oui
Confidentialité	Non	Oui
Intégrité	Non	Oui
Authentification	Non	Oui
Utilisation actuelle	Rare	Standard

A noter
CIA
Confidentialité (chiffrement)
Intégrité (données non modifiées)
Authentification (certificat)

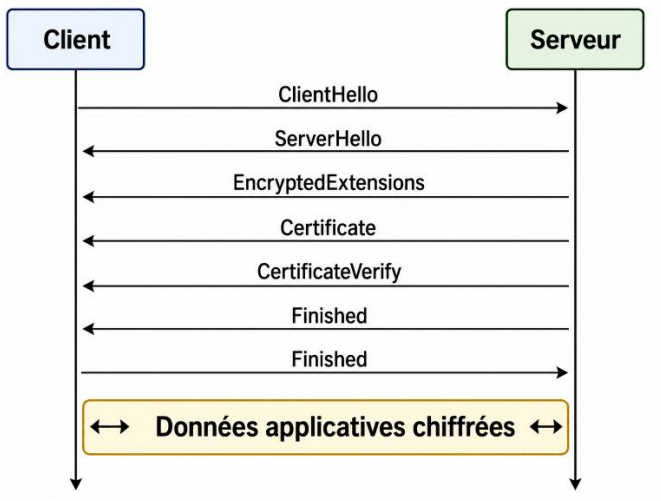
Protocole TLS version 1.3

TLS signifie Transport Layer Security et est le successeur de SSL (Secure Sockets Layer).

TLS assure une communication sécurisée entre les navigateurs Wb et les serveurs (chiffrement avec clés uniques symétriques).

TLS établit la communication entre le serveur et le client, détermine le type de chiffrement et les clés à utiliser, et vérifie l'intégrité des données transmises, garantissant ainsi que les informations sensibles ne peuvent pas être consultées ou interceptées par une autre partie.

Handshake (poignée de main) du protocole TLS version 1.3



1 – **Client Hello** (Proposer une méthode de communication sécurisée)
Le client (navigateur) contacte le serveur et envoie :

- les versions TLS qu'il supporte
- les algorithmes de chiffrement qu'il accepte
- une valeur aléatoire (Client Random)
- sa clé publique éphémère
- le nom du site demandé (pas obligatoire)

La clé publique est très importante : elle sert à préparer la création d'une clé secrète commune entre le client et le serveur.

2 – **Serveur Hello** (Réponse du serveur)
Le serveur choisit les paramètres définitifs :

- la version TLS choisie
- l'algorithme retenu
- une valeur aléatoire (Server Random)
- sa clé publique éphémère

A partir de maintenant les messages suivants sont chiffrés.

3 – **EncryptedExtensions**
Le serveur envoie des paramètres supplémentaires liées au protocole applicatif : HTTPS ou SMTPS ou …

4 – **Certificat**
Le serveur envoie son certificat pour prouver son identité.
Le client vérifie :

- Que le certificat est valide
- Qu'il correspond bien au site demandé
- Qu'il est signé par une autorité de confiance

Remarque
C'est cette étape qui permet d'éviter qu'un faux serveur se fasse passer pour le vrai site. Le client calcule de son coté la clé secrète, le serveur calcule aussi sa clé secrète. Ces clés ne sont jamais transmises sur le réseau.

5 – **CertificateVerify**
Le serveur envoie ensuite CertificateVerify. Ce message sert à prouver que le serveur possède bien la clé privée correspondant au certificat.

6 – **Finished** du serveur
Le serveur envoie ensuite un message **Finished**. Ce message prouve que le serveur a bien calculé les bonnes clés de session et que le handshake n'a pas été modifié.

7 – **Finished** du client
Le client répond aussi avec son message **Finished**.

À partir de ce moment, le handshake TLS 1.3 est terminé. Le client et le serveur commencent une session chiffrée.