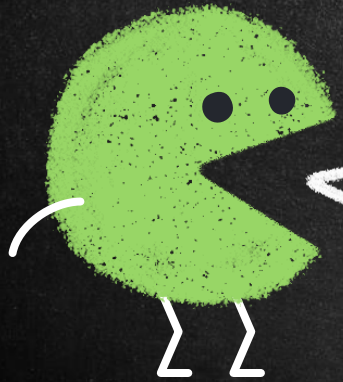
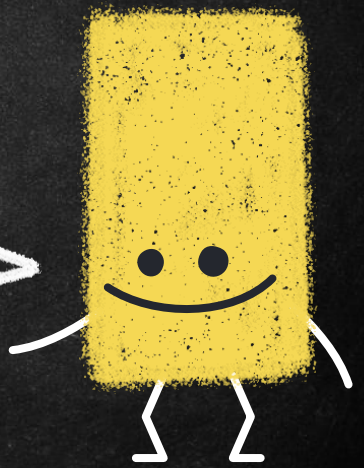
Sécurité des données

A2021

HTTPS

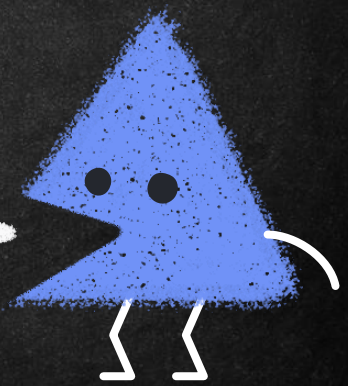




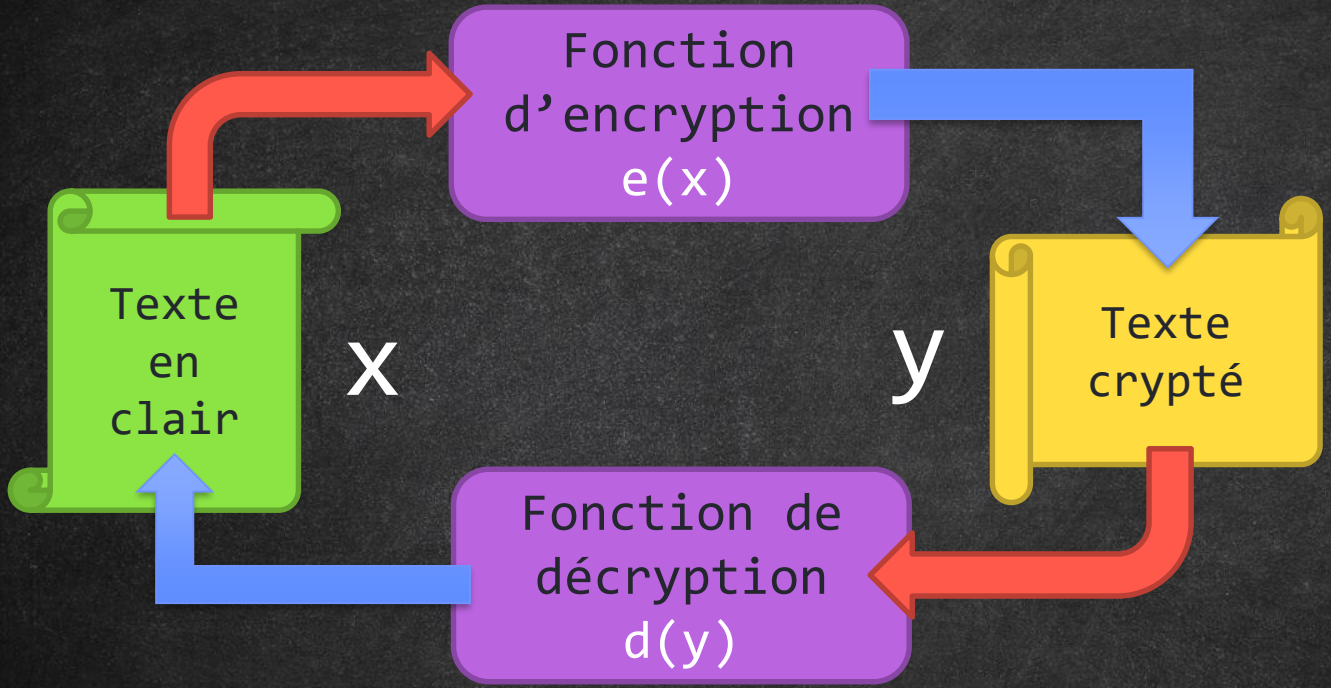
Yannick Charron

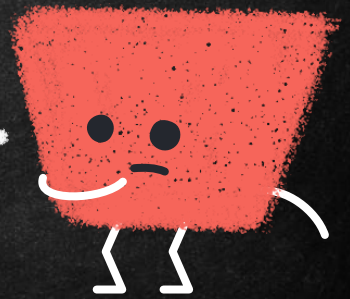
Plan de la séance

- Retour - Chiffrement asymétrique
- Signature
- HTTP
- HTTPS
  - SSL / TLS



Concept Général





Récupération de la clé publique du destinataire

Clé publique de Garfield

Clé privée de Garfield

Chiffrement du message avec la clé publique du destinataire

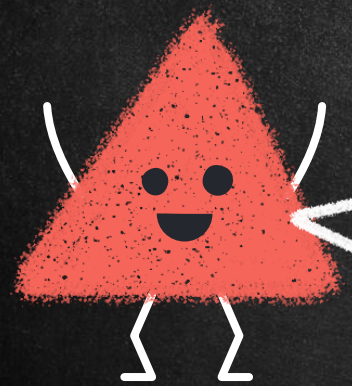
Lasagne aux kiwis

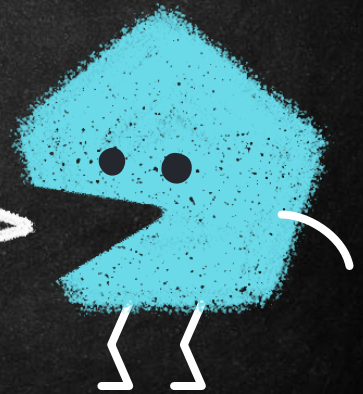
0xAk9uiVZ3

0xAk9uiVZ3

Lasagne aux kiwis

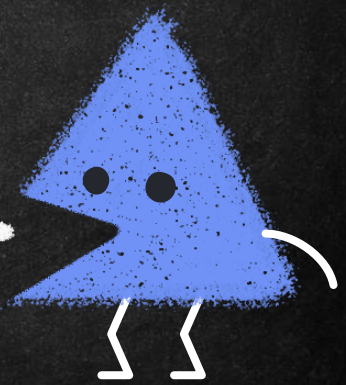
Signature





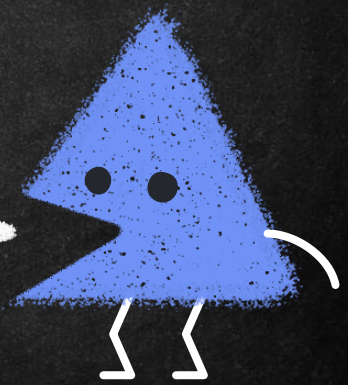
Valider authenticité

- Étant donné que nous utilisons la clé publique pour chiffrer le message
- Il est possible de tenter d'envoyer un message comme si nous étions quelqu'un d'autre
- Le destinataire ne peut valider l'authenticité du message



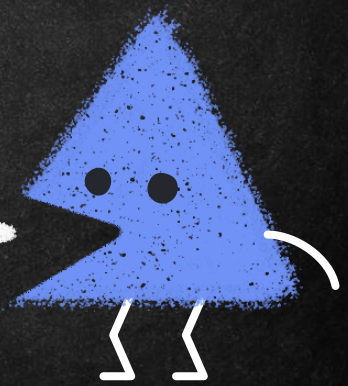
Signature symétrique

- HASH
- HMAC = Hash + Salt (partagé)



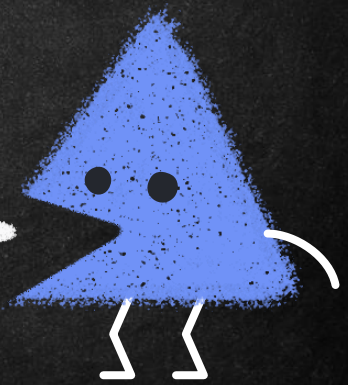
Signature asymétrique

- Utilise un algorithme de *hash* pour mettre le message à signer dans une taille fixe
- Signe le message avec une clé privée
- La clé publique pourra vérifier la signature

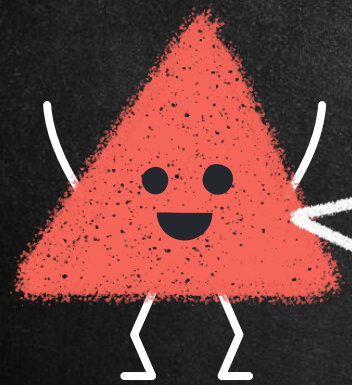


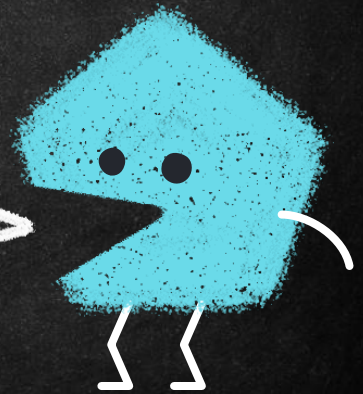
Chiffrement Double

- Chiffrer avec sa clé privée + chiffre le résultat avec la clé publique du destinataire
- Il doit déchiffrer avec sa clé privée et ensuite par votre clé publique



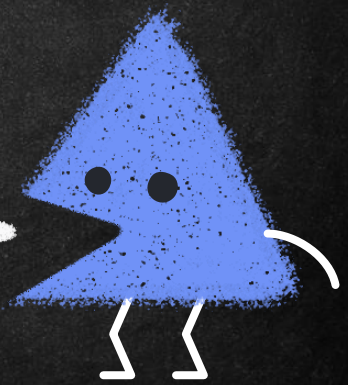
HTTPS





Vocabulaire

- SSL et TLS
- Handshake
- Cipher Suites
- Key Exchange
- Certificates
- Certificates Authorities
- Trusted Certificates Authorities



SSL

- Secure Socket Layer
- Netscape en 1995
- 3 versions
- Toutes les versions sont vulnérables et sont maintenant *obsolètes*

TLS

- Transport Layer Security
- IETF en 1999
- 4 versions: 1.0, 1.1, 1.2 et 1.3
- TLS 1.0 et 2.0 vulnérables
- Version 1.2 est la plus couramment utilisée
- Le successeur de SSL

On ne devrait plus dire SSL, mais ...

Transport Layer Security

- 3 objectifs
- Échanger un ensemble d'algorithmes de chiffrements (Cipher suites) et ses paramètres
- Authentifier une ou les deux parties de la communication
- Créer et échanger une clé de session symétrique

Négocier une Cipher suites

- Chaque client et serveur ont des caractéristiques différentes et des paramètres de personnalisation.
- Le client et le serveur partagent leurs capacités cryptographiques
- Pour se mettre d'accord sur les caractéristiques (algorithmes et paramètres) cryptographiques communes à utiliser.
- L'ensemble des caractéristiques se nomme une *Cipher suite*.

Négocier une Cipher suites

`TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`

- TLS est le protocole
- ECDHE est l'algorithme d'échange de clé
- ECDSA est l'algorithme d'authentification
- AES 128 GCM est l'algorithme de chiffrement symétrique
- SHA256 est l'algorithme de hash

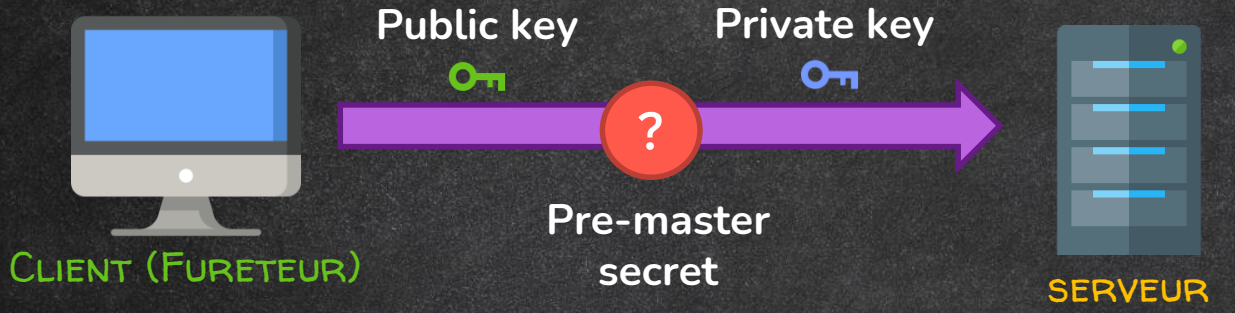
Authenticité

- S’assurer que l’autre partie prenante est bel et bien celle qu’elle dit être
- Via un certificat ou une chaine de certificat
- Une autorité de confiance doit signer le certificat

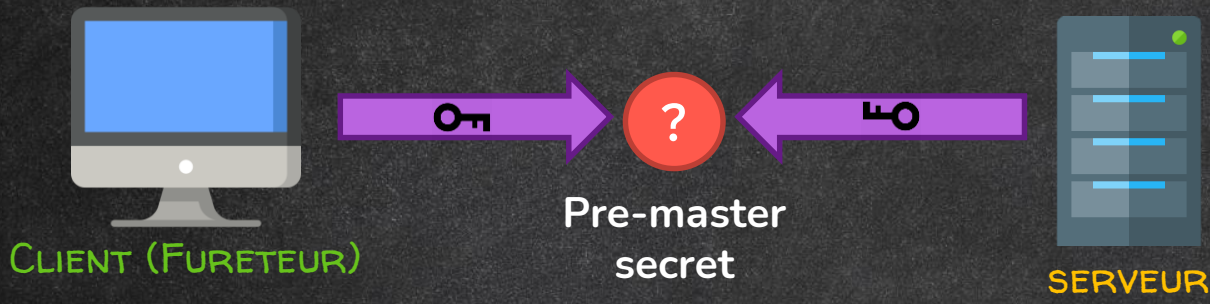
Échange d'une clé

- Partager une clé de chiffrement unique
  - Seulement le client et le serveur connaissent cette clé
- La communication utilisera cette clé symétrique pour le reste de la communication

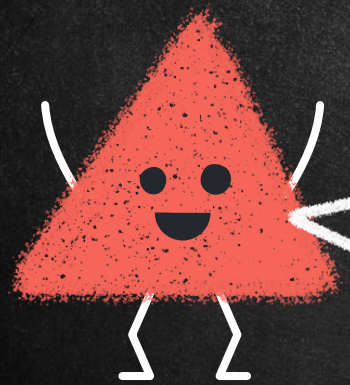
Échange d'une clé - RSA

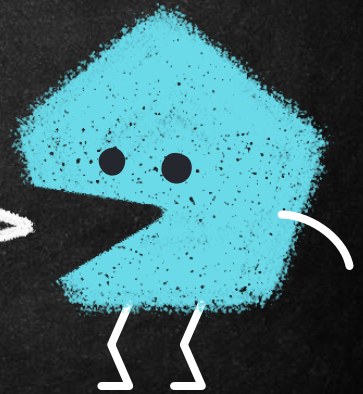


Échange d'une clé – Diffie-Hellman



Hand Shake





CLIENT (FURETEUR)

Client Hello

Server Hello

Certificate

Server Key Exchange

Server Hello Done

Client Key Exchange

Change Cipher Spec

Finished

Change Cipher Spec

Finished

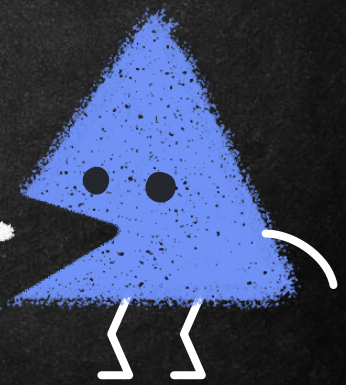
SERVEUR

Client Hello

Client Hello

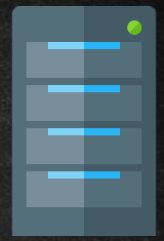
- Le client informe le serveur de ses capacités cryptographiques (*Cipher Suites*)
- Inclus également un long nombre premier (*client random*)

Client (Fureteur)

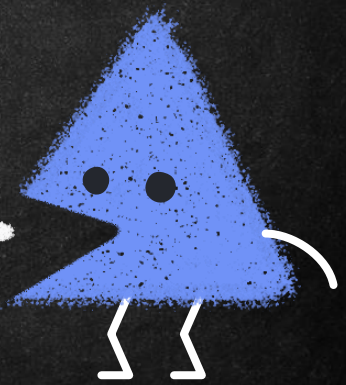


Server Hello

- Le serveur informe le client du choix de la *Cipher Suite*
- Le serveur transmet aussi un grand nombre premier (*server random*) au client

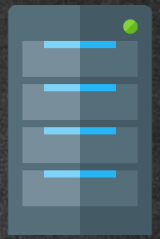


SERVEUR



Étape 1 – Terminée

Client (Fureteur)



Serveur

Key	Cipher	Hash
RSA	RC4	HMAC-MD5
Diffie-Hellman	Triple DES	HMAC-SHA
DSA	AES	SHA384

Version	V3
Client random	5b 56 f3 8a cc ae 8f 94 94 3f
Server random	ae 5a 21 6c 3c 81 8a 80 7d 33
Pre-master secret	
Session key	

Key	Cipher	Hash
RSA	RC4	HMAC-MD5
Diffie-Hellman	Triple DES	HMAC-SHA
DSA	AES	SHA384

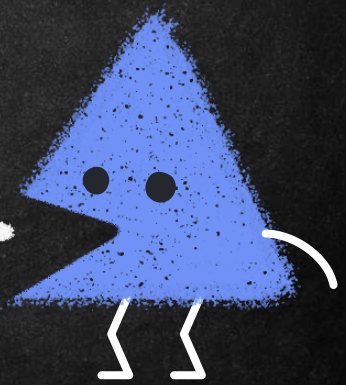
Version	V3
Client random	5b 56 f3 8a cc ae 8f 94 94 3f
Server random	ae 5a 21 6c 3c 81 8a 80 7d 33
Pre-master secret	
Session key	

Certificate

Certificate

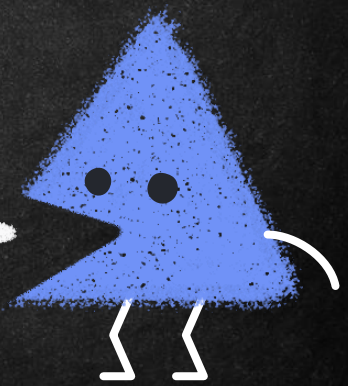
- Le serveur transmet son certificat TLS (certificat chain) au client
- Le client démarre alors le processus de validation du certificat

SERVEUR

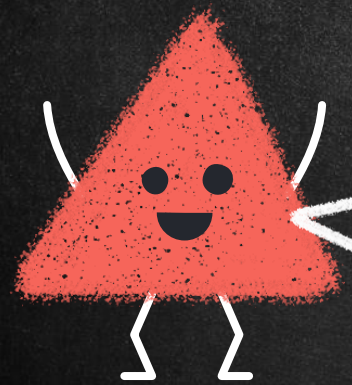


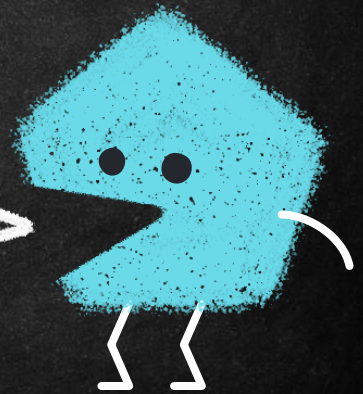
Certificate

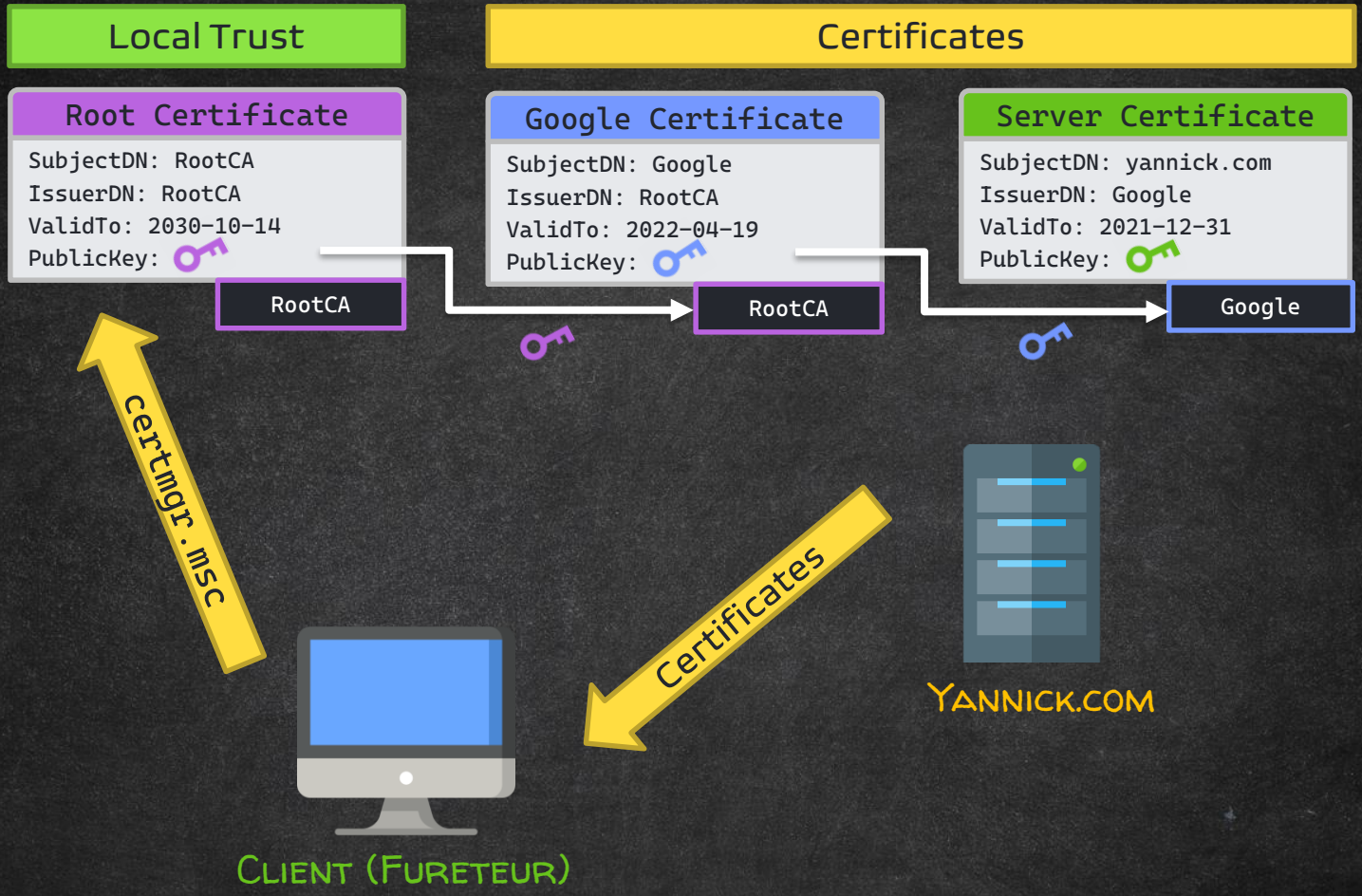
- Informations comprises dans le certificat
  - Nom courant (domaine)
  - Émetteur
  - Date de validité
  - Clé publique
- Généralement le certificat est composé d'une chaine de certificats
- Allons voir dans un fureteur



Validation du Certificat



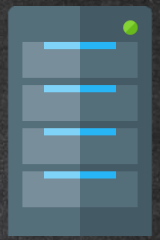



Local Trust
Certificates
Root Certificate
SubjectDN: RootCA
IssuerDN: RootCA
ValidTo: 2030-10-14
PublicKey:
RootCA
Google Certificate
SubjectDN: Google
IssuerDN: RootCA
ValidTo: 2022-04-19
PublicKey:
RootCA
Server Certificate
SubjectDN: yannick.com
IssuerDN: Google
ValidTo: 2021-12-31
PublicKey:
Google
certmgr.msc
Certificates
Client (Fureteur)
Yannick.com

Étape 2 – Terminée

Client (Fureteur)

J'ai maintenant confiance avec qui je communique



SERVEUR

Key	Cipher	Hash
RSA	RC4	HMAC-MD5
Diffie-Hellman	Triple DES	HMAC-SHA
DSA	AES	SHA384

Version	V3
Client random	5b 56 f3 8a cc ae 8f 94 94 3f
Server random	ae 5a 21 6c 3c 81 8a 80 7d 33
Pre-master secret	
Session key	

Key	Cipher	Hash
RSA	RC4	HMAC-MD5
Diffie-Hellman	Triple DES	HMAC-SHA
DSA	AES	SHA384

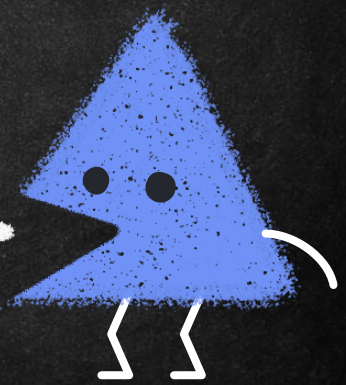
Version	V3
Client random	5b 56 f3 8a cc ae 8f 94 94 3f
Server random	ae 5a 21 6c 3c 81 8a 80 7d 33
Pre-master secret	
Session key	

Server Key Exchange

Server Key Exchange

- Étape optionnelle en fonction méthode d'échange de clé (Diffie-Hellman)
- Le serveur doit fournir de l'information supplémentaire pour l'échange de clé

SERVEUR

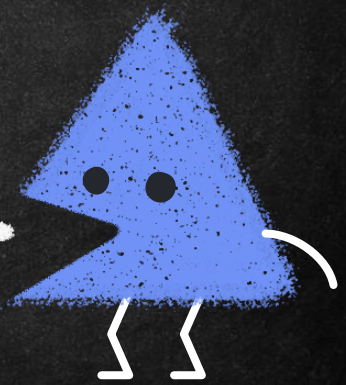


Server Hello Done

Server Hello Done

- Le serveur a transmis l'ensemble de ses messages, il est en attente du client.

SERVEUR

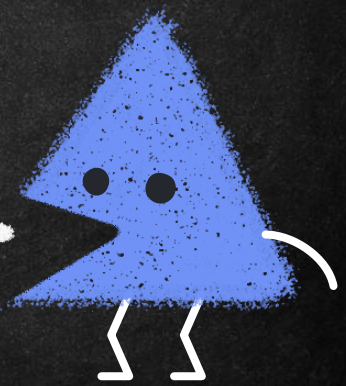


Client Key Exchange

Client Key Exchange

- Le client doit fournir l'information pour l'échange de clé.
- L'information de l'algorithme retenu
  - RSA = pre-master secret (chaine aléatoire)
- L'information est chiffrée avec la clé publique du serveur et transmise au serveur

Client (Fureteur)

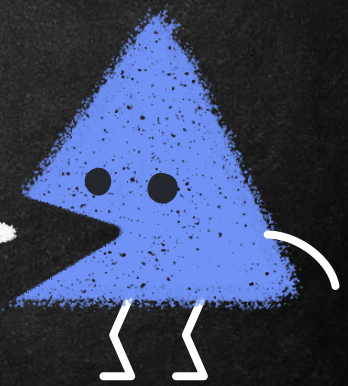


Change Cipher Spec

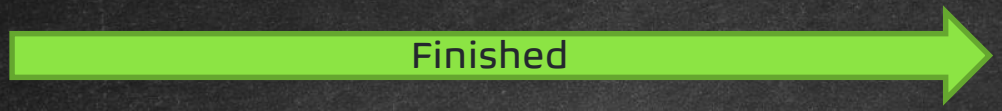
Change Cipher Spec

- Le client génère la clé session (symetric) avec
  - *client random*
  - *server random*
  - *pre master secret*
- Informe le serveur qu'il a généré la clé de session et qu'il changera au mode de communication chiffrée symétrique

Client (Fureteur)

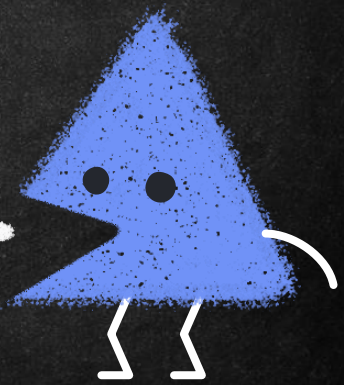


Finished



- Le client envoie le message Finished en utilisant la clé de session symétrique
- Ceci est le premier message chiffré avec la clé de session symétrique
- Le message contient un hash pour valider l'intégrité du message

Client (Fureteur)

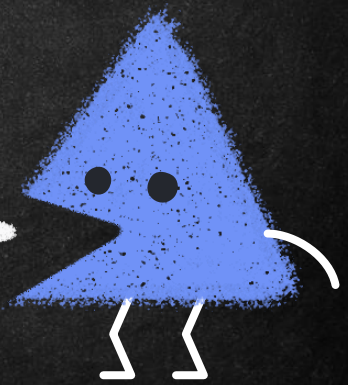


Change Cipher Spec

Change Cipher Spec

SERVEUR

- Le serveur déchiffre avec sa clé privée, le *pre-master secret*
- Le serveur génère la clé session (symetric) avec
  - *client random*
  - *server random*
  - *pre master secret*
- Informe le client qu'il a généré la session (symetric) key et qu'il changera au mode de communication chiffrée symétrique

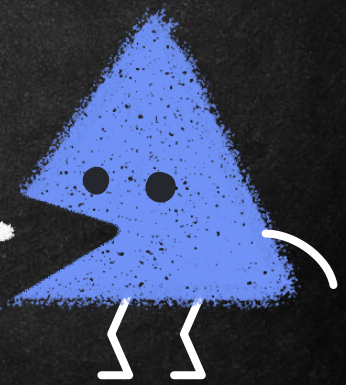


Finished

Finished

- Le serveur envoie le message Finished en utilisant la clé de session symétrique
- Le message contient un *hash* pour valider l'intégrité du message

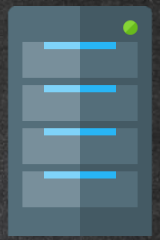
SERVEUR



Étape 3 – Terminée

Client (Fureteur)

J'ai maintenant confiance avec qui je communique



SERVEUR

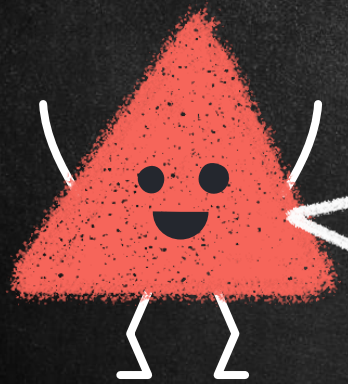
Key	Cipher	Hash
RSA	RC4	HMAC-MD5
Diffie-Hellman	Triple DES	HMAC-SHA
DSA	AES	SHA384

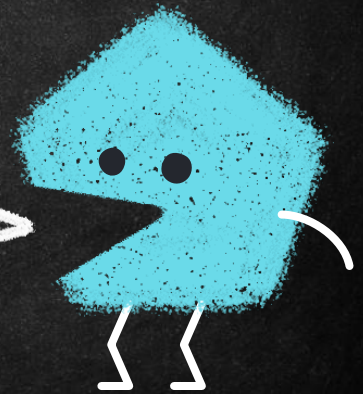
Version	V3
Client random	5b 56 f3 8a cc ae 8f 94 94 3f
Server random	ae 5a 21 6c 3c 81 8a 80 7d 33
Pre-master secret	e7 7e db f4 cd cf ab 30 a3 0d
Session key	8c 67 72 94 74 13 b9 d5 07 5f

Key	Cipher	Hash
RSA	RC4	HMAC-MD5
Diffie-Hellman	Triple DES	HMAC-SHA
DSA	AES	SHA384

Version	V3
Client random	5b 56 f3 8a cc ae 8f 94 94 3f
Server random	ae 5a 21 6c 3c 81 8a 80 7d 33
Pre-master secret	e7 7e db f4 cd cf ab 30 a3 0d
Session key	8c 67 72 94 74 13 b9 d5 07 5f

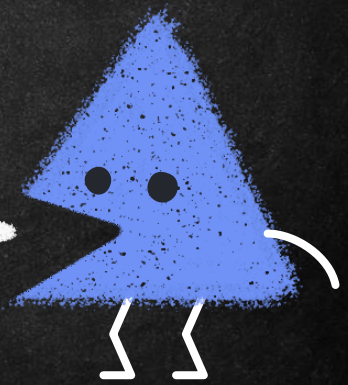
La communication chiffrée et authentifiée commence





TLS 1.3

- L’objectif premier était d’optimiser le nombre d’aller-retour entre le client et le serveur
- Le rendre encore plus sécuritaire
- Offrir moins de *Cipher Suites* de 37 à 5
  - *Négociation plus simple*
- Des étapes du *handshaking* sont fusionnées



1.2

```
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
```

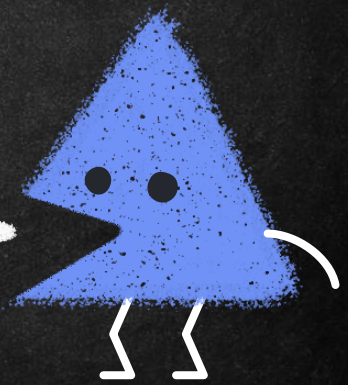
1.3

```
TLS_AES_256_GCM_SHA384
```

```
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256
TLS_AES_128_GCM_SHA256
TLS_AES_128_CCM_8_SHA256
TLS_AES_128_CCM_SHA256
```

Génération d'un certificat

- Self-Signed
- Via un Certificates Authority
  - Let's encrypt



WireShark