

# Diagramme de séquence de l'établissement d'une session SSH

```
<uml> @startuml
group Etablissement d'une session 3-Way Handshake
ClientSSH → Serveur : demande session avec paquet SYN (Synchronize)
Serveur → ClientSSH : approuve connexion avec paquet SYN-ACK (acknowledgement)
ClientSSH → Serveur : confirme avec paquet ACK
end group
Negociation des algorithmes de chiffrement
ClientSSH → Serveur : annonce la version de SSH utilisée
Serveur → ClientSSH : annonce la version de SSH utilisée
ClientSSH → ClientSSH : génèrent des paires de clés \n publiques-privées temporaires
Serveur → Serveur : génèrent des paires de clés \n publiques-privées temporaires
ClientSSH → Serveur : Initialisation échange de clé\n(Key Exchange Init)\n+ paramètres (algorithmes ; compression)
Serveur → ClientSSH : réponse à l'initialisation de l'échange de clé\n(Key Exchange Init)\n+ paramètres (algorithmes ; compression)
ClientSSH → Serveur : Initialisation d'échange de clés Diffie-Hellman\n(Elliptic Curve Diffie-Hellman Key Exchange Init)
Serveur → Serveur : calcul de la clé symétrique
Serveur → ClientSSH : Réponse d'échange de clés Diffie-Hellman\n+ informe nouvelle clé symétrique calculée\n(Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys)
ClientSSH → ClientSSH : calcul de la clé symétrique
Serveur → ClientSSH : informe nouvelle clé symétrique calculée
end group
Etablissement d'un canal sécurisé
Serveur → ClientSSH : envoi clé publique + empreinte du serveur (fingerprint)
ClientSSH → ClientSSH : Vérification de l'identité du serveur \nEtablissement canal sécurisé
```

```
end group
Authentification par mot de passe
Serveur → ClientSSH : demande saisie login
ClientSSH → Serveur : envoi login
Serveur → ClientSSH : demande saisie mot de passe
ClientSSH → Serveur : envoi mot de passe
Serveur → Serveur : vérification login et mot de passe
end @enduml
</uml>
```

## Compléments

- [Les connexions TCP](#)
- L'algorithme **chacha20\_poly1305** ([RFC 7539](#))

L'algorithme de chiffrement symétrique **ChaCha20** est utilisé à la place de **AES 256**.

- **ChaCha20** : algorithme de chiffrement symétrique plus rapide qu'AES sur un matériel générique (mise en œuvre purement en logiciel)
- **Poly1305** : c'est un **MAC** (message authentication code) qui permet d'assurer l'**intégrité des données** en vérifiant qu'elles n'ont subi aucune modification après une transmission. C'est une fonctionnalité semblable à une fonction de hachage.

La combinaison de ces deux algorithmes permet de faire du chiffrement intègre.

## Ressources

- <https://www.bortzmeyer.org/7539.html>
- [https://fr.wikipedia.org/wiki/Code\\_d%27authentification\\_de\\_message](https://fr.wikipedia.org/wiki/Code_d%27authentification_de_message)
- <https://serverfault.com/questions/586638/understand-wireshark-capture-for-ssh-key-exchange>
- <https://serveur.ipgirl.com/comprendre-la-capture-wireshark-pour-lchange-de-keys-ssh.html>

From:  
/ - **Les cours du BTS SIO**

Permanent link:  
</doku.php/reseau/debian/sequencesessionssh?rev=1684686141>

Last update: **2023/05/21 18:22**

