

# 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
Etablissement de la session
ClientSSH → Serveur : annonce la version de SSH utilisée
Serveur → ClientSSH : annonce la version de SSH utilisée
Serveur → ClientSSH : envoie sa clé publique
ClientSSH → ClientSSH : génèrent paires de clés (privées - publique) temporaires
Serveur → Serveur : génèrent paires de clés (privées - publique) temporaires
ClientSSH ↔ Serveur : algorithme d'échange de clés Diffie-Hellman pour déterminer la clé partagée (symétrique)
note left : clé de session symétrique
note right : clé de session symétrique
ClientSSH → ClientSSH : Vérification de l'identité du serveur
Acceptation de l'identité du serveur
end group
Etablissement d'un canal sécurisé
end group
Authentification par mot de passe
ClientSSH → Serveur : demande saisie login
ClientSSH → Serveur : envoi login
Serveur → Serveur : demande saisie mot de passe
ClientSSH → Serveur : envoi mot de passe
Serveur → Serveur : vérification login et mot de passe
Serveur → ClientSSH : affichage du prompt
end
  
```

```

group Authentification par clé SSH
ClientSSH → Serveur : fait enregistrer sa clé publique dans authorizedkeys
Serveur → ClientSSH : demande saisie login
ClientSSH → Serveur : envoi login
Serveur → Serveur : envoie un défi (challenge)
ClientSSH → ClientSSH : signe le défi avec sa clé privée
ClientSSH → Serveur : envoi réponse au challenge
Serveur → Serveur : vérifie la signature avec la clé publique enregistrée dans authorizedkeys
Serveur → ClientSSH : affichage du prompt
end
  
```

group Communication chiffrée entre le client et le serveur durant toute la session

```

end group
Deconnexion
ClientSSH → Serveur : déconnexion avec paquet RST,ACK (Reset)
Serveur → ClientSSH : approuve déconnexion avec paquet ACK
end @enduml
  
```

## 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://fr.wikipedia.org/wiki/%C3%89change\\_de\\_cl%C3%A9s\\_Diffie-Hellman](https://fr.wikipedia.org/wiki/%C3%89change_de_cl%C3%A9s_Diffie-Hellman)
- <https://www.adikts.io/mieux-comprendre-ssh-et-lauthentification-par-cles/>
- <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=1749582656](https://doku.php/reseau/debian/sequencesessionssh?rev=1749582656)

Last update: **2025/06/10 21:10**

