# SSH simplifié

## Introduction

Secure Shell (SSH) est à la fois un **programme informatique** et un **protocole de communication sécurisé**.

Le protocole de connexion **impose un échange de clés de chiffrement** en début de connexion.   
Par la suite, **tous les segments TCP sont authentifiés et chiffrés**. Il devient donc impossible d'utiliser un renifleur comme Wireshark pour voir ce que fait l'utilisateur.

Habituellement le protocole SSH utilise le **port TCP 22**.

## 

## openSSH

Le **paquet le plus répandu** utilisé sous Linux pour fournir des connexions rapides et sécurisées aux serveurs Linux est **openSSH**.   
Le paquet openSSH utilise le protocole Secure Shell (SSH) pour **établir une connexion chiffrée entre deux périphériques réseau**.

L'installation du serveur oenSSH crée un **répertoire /etc/ssh** et génère un couple de clés RSA …

* **/etc/ssh/ssh\_host\_rsa\_key** contenant la clé privée du serveur ;
* /**etc/ssh/ssh\_host\_rsa\_key.pub** contenant la clé publique du serveur.

## Installation et configuration de base SSH

### Installation du serveur SSH

**Remarque** …  
Le client SSH est disponible dans le **paquet openssh-client**, qui est, pour la **plupart des distributions, préinstallé**.

Afin de **pouvoir établir une connexion à distance**, on doit installer le service SSH …  
**>> sudo apt install openssh-server**

L'installation comporte une **étape de génération des clefs de chiffrement**.  
Par la suite, le service SSH est lancé.

## Établissement d'une connexion SSH

Les étapes de l'établissement d'une connexion SSH sont …

1. Le **serveur** **envoie sa clef publique au client**.   
   Celui-ci **vérifie qu'il s'agit bien de la clef du serveur**, s'il l'a déjà reçue lors d'une connexion précédente ;
2. Le **client génère une clef secrète** (chiffrement symétrique) et l'envoie au serveur, en chiffrant cet échange avec la clef publique du serveur (chiffrement asymétrique).   
   Le **serveur déchiffre cette clef secrète en utilisant sa clé privée**, ce qui confirme qu'il est bien le vrai serveur ;
3. Afin de le prouver au client, il **chiffre un message standard avec la clef secrète et l'envoie au client**.  
   Si le client **retrouve le message standard en utilisant la clef secrète**, il a la preuve que le **serveur est bien le vrai serveur** ;
4. Une fois la **clef secrète échangée**, le **client et le serveur peuvent alors établir un canal sécurisé** grâce à la clef secrète commune (chiffrement symétrique) ;
5. Une fois que le **canal sécurisé est en place**, le client va pouvoir envoyer au serveur une demande d’ouverture de session et le mot de passe de l'utilisateur pour vérification.   
   Le **canal sécurisé reste en place jusqu'à ce que l'utilisateur se déconnecte**.

## Première connexion avec SSH Authentification par mot de passe

L’**authentification** **par mot de passe** est la **méthode la plus simple**.

Depuis l’hôte client …  
**>> ssh <Identifiant>@<Nom DNS ou Adresse IP du serveur>  
>> ssh tux@srv01.profsavard.info**

Si le **même identifiant est utilisé** sur le client et sur le serveur …  
**>> ssh <Nom DNS ou Adresse IP du serveur>  
>> ssh srv01.profsavard.info**

Si on doit utiliser un port différent …  
**>> ssh <Nom DNS ou Adresse IP du serveur> -p <Port>  
>>** **ssh tux@srv01.profsavard.info -p 12345**L'option -p <Port>, qui précise le port utilisé par le serveur, est facultative.   
Si rien n'est précisé, c'est le **port TCP 22** qui sera utilisé par défaut.

Pour se connecter avec SSH en IPV6 depuis un terminal ...  
**>> ssh -6 <Adresse IPv6 du serveur>  
>>** **ssh -6 tux@2a01:e35:2431::2e57**

**Remarque** ...  
Afin de pouvoir se connecter en IPV6, il faut que le serveur SSH puisse écouter les adresses IPV6.

Pour ce faire, il faut ajouter le code suivant dans le **fichier /etc/ssh/sshd\_config** sur le serveur ...  
**ListenAddress ::**

## Configuration du serveur SSH

Le **fichier de configuration du serveur SSH est /etc/ssh/sshd\_config**.

**Remarque** …  
Ce fichier ne doit pas être confondu avec le **fichier /etc/ssh/ssh\_config**, qui est le **fichier de configuration du client SSH**.

En règle générale, les programmes client et serveur openSSH fonctionnent correctement tels quels après l’installation.   
Toutefois, il peut arriver que l’on souhaite personnaliser leur comportement.

Le **répertoire /etc/ssh** est **l'emplacement de stockage des fichiers de configuration** Secure Shell.   
Le **fichier de configuration /etc/ssh/sshd\_conf** définit les options du serveur au format …

**Directive Valeur**

L'**entrée directive** est le nom d'une directive à définir et l’**entrée valeur** est la valeur affectée à cette option.

Voici les **directives les plus importantes** de ce fichier de configuration …

**Port**Cette directive indique que le **serveur SSH écoute sur le port 22**, qui est le port par défaut de SSH.   
il est **possible de faire écouter le service sur un autre port** en modifiant la valeur de cette directive.   
On peut aussi le faire **écouter sur plusieurs ports à la fois** en ajoutant des lignes similaires.  
Il est possible de modifier le serveur SSH pour écouter un autre port à l'aide de la directive Port…  
**Port 2096**

**ListenAddress**Dans certains cas, on peut avoir plusieurs interfaces réseau (ou interfaces virtuelles) et on aimerait limiter le serveur SSH afin qu'il n'écoute que certaines des cartes réseau.   
Pour ce faire, on utilise la directive ListenAddress et on spécifie l'adresse (ou les adresses) IP attribuée(s) aux interfaces réseau sur lesquelles SSH doit accepter les connexions …  
**ListenAddress 192.168.1.100:192.168.1.101**

**Attention** ...  
Si on a modifié le fichier de configuration du serveur, il faut indiquer au service de relire son fichier de configuration …  
**>> sudo systemctl reload|restart sshd.service**

### Directives qui affectent l'accès utilisateur

Il est possible d’envisager de **ne pas autoriser l'utilisateur root à se connecter** directement à l’aide du protocole SSH, en particulier si le **système est connecté à Internet**. **Autoriser la connexion root** à l’aide du protocole SSH **offre aux pirates potentiels** la possibilité d’utiliser une **attaque par force brute** en essayant plusieurs mots de passe pour obtenir un accès.

Un meilleur choix consiste à **autoriser une connexion utilisateur régulière**, permettant à l'utilisateur de **basculer vers le compte root à l'aide de la commande su**. Même si l’attaque par force brute peut toujours être tentée par l’utilisateur habituel, le pirate doit savoir que le compte existe et, après avoir obtenu l’obtention d’une tentative d’accès, doit savoir le mot de passe de l’utilisateur root.

**Directive PermitRootLogin**

**Remarque** …  
Cette **directive est commentée** par défaut.

Cette directive, qui est **définie par défaut avec la valeur prohibit-password**, interdit toutes les méthodes d'authentification interactives, **n'autorisant que celle utilisant une clé publique**.   
Les valeurs peuvent être **yes**, **prohibit-password**, **forced-commands-only**, ou **no**.

Si cette option est **définie sur prohibit-password** (ou son alias obsolète without-password), le **mot de passe et l'authentification** **interactive au clavier sont désactivés** pour l’utilisateur root.

Si cette option est définie sur **forced-commands-only,** une connexion root avec **authentification par clé publique est autorisée**, mais uniquement **si l'option de commande a été spécifiée** (ce qui peut être utile pour effectuer des sauvegardes à distance même si la connexion root n'est normalement pas autorisée). Toutes les **autres méthodes d'authentification sont désactivées** pour l’utilisateur root.

Si cette option est définie sur **no**, root n'est pas autorisé à se connecter.

Si cette **directive est égale à no**, ce qui signifie que root ne peut se connecter à distance, on doit d'abord vous **connecter par SSH en tant que simple utilisateur**, puis utiliser la commande su pour **basculer à root**. Sans cela, un pirate n'aurait qu'à trouver le mot de passe du compte root, alors que là, il doit trouver un compte d’utilisateur et son mot de passe.

**Directive AllowUsers et DenyUsers**

Il est aussi possible de limiter seulement certains utilisateurs pouvant se connecter à l’aide de SSH  
Dans ce cas, il existe deux choix … **AllowUsers** ou **DenyUsers**.

Ces deux mots clés vous permettent de spécifier une liste de noms d'utilisateurs …  
**AllowUsers tux kermit squiddly**

* Si on utilise le **directive AllowUsers**, seuls ces utilisateurs seront autorisés à se connecter au serveur SSH ;
* Si on utilise le **directive DenyUsers**, tous les utilisateurs peuvent se connecter au serveur SSH, à l'exception des utilisateurs répertoriés.

**Remarque** …  
Si on utilise les deux directives, **DenyUsers est prioritaire**.

Il est possible d’utiliser des caractères génériques pour faire correspondre les modèles …  
**DenyUsers invit\* # Refuser à tout utilisateur portant un nom commençant par « invit »  
DenyUsers ????app # Refuser à tout utilisateur avec un nom qui contient 4 caractères et qui se termine par « app »**

**Remarque** …  
Il existe également une **directive AllowGroups** et une **directive DenyGroups**.

Le tableau suivant répertorie les **diectives les plus courantes** que l’on devra peut-être modifier sur le serveur openSSH.

|  |  |
| --- | --- |
| Directive | Description |
| Banner | Spécifie le fichier à afficher avant la connexion d’un usager |
| AllowUsers | Autorise la liste d'utilisateurs spécifiée à se connecter à l'aide de SSH |
| DenyUsers | Empêche la liste d'utilisateurs spécifiés de se connecter à l'aide de SSH |
| PasswordAuthentication | Autorise l'authentification à l'aide de mots de passe textuels |
| MaxAuthTries | Spécifie le nombre de tentatives qu’un utilisateur peut entrer un mot de passe (défaut 6) |
| PermitRootLogin | Autorise le compte utilisateur root à se connecter à l'aide de SSH |
| PermitEmptyPassword | Spécifie si un usager peut se connecter sans mot de passe (défaut No) |
| PrintMotd | Spécifie si le contenu du fichier /etc/motd est affiché lorsqu’un usager se branche avec SSH |
| Protocol | Spécifie le niveau de protocole de chiffrement – Le niveau 2 est préféré et plus sécurisé |
| PubkeyAuthentication | Autorise l'authentification à l'aide de certificats |
| X11Forwarding | Autorise les serveurs X distants à exécuter des applications client X à l'aide d'un tunneling |
| AllowTcpForwarding | Autorise le serveur à accepter les protocoles en tunnel |
| listenAddress | Spécifie les interfaces qu’écoute SSH |
| Port | Modifie le serveur SSH pour écouter un autre port |

La plupart de ces directives prennent une **simple valeur oui ou non** pour activer ou désactiver la fonctionnalité.   
Les **directives AllowUsers et DenyUsers** utilisent une **liste de noms de compte d'utilisateur** sur le système, **séparés par des espaces**, afin de déterminer les utilisateurs à autoriser ou à bloquer l'accès au serveur.

Le **fichier de configuration sshd\_conf** par défaut **contient de nombreuses autres options**, décrites dans les commentaires qu'il contient. Il est important de de prendre le temps de parcourir chaque option pour déterminer s’il s’agit d’un avantage pour son environnement réseau.

Après avoir **apporté des modifications à la configuration**, il faudra **redémarrer le serveur openSSH** pour que celles-ci prennent effet.   
Il est également possible de **simplement recharger le fichier de configuration**, cela permettra de ne pas déconnecter les utilisateurs utilisant le service.

Pour ce faire, on utilise …  
**>> sudo systemctl reload|restart sshd.service**

## Utilisation d’openSSH

Il est intéressant de connaître plusieurs commandes de client SSH …

* **ssh**  
  Cette commande permet de se connecter à un serveur SSH et de travailler sur le serveur à l’aide d’une ligne de commande ;
* **scp**  
  Cette commande permet de transférer des fichiers vers et/ou depuis un serveur SSH à l’aide de la ligne de commande ;
* **sftp**  
  Cette commande permet de se connecter à un serveur SSH et d'exécuter des commandes de type **ftp**.

On doit **connaître la fonction** de chacune de ces commandes ainsi que certaines **directives clés du fichier /etc/ssh/ssh\_config**.

### Fichier ssh\_config

Quelques **directives du fichier ssh\_config sont différentes du fichier sshd\_config** (voir plus loin).   
Pour commencer, il y a le **fichier système /etc/ssh/ssh\_config**, qui s’applique à **tous les utilisateurs**.   
De plus, **chaque utilisateur peut créer un fichier dans son répertoire personnel** (~/.ssh/config), qui peut être utilisé pour remplacer les directives du fichier /etc/ssh/ssh\_config.

Outre les fichiers, les **options de ligne de commande peuvent remplacer les valeurs spécifiées** dans les fichiers de configuration.   
L'ordre dans lequel toutes ces informations sont analysées est …

* **Options de ligne de commande** ;
* **Fichier ~/.ssh/config de l'utilisateur** ;
* **Fichier /etc/ssh/ssh\_config** (fichier global).

**Important** …  
La **première directive trouvé est celle utilisée.**   
Par exemple, si **ConnectTimeout** est défini dans le fichier ~/.ssh/config de l'utilisateur et qu'une valeur différente est définie dans le fichier /etc/ssh/ssh\_config, le fichier de configuration de l'utilisateur est utilisé pour définir cette valeur.

Une autre **différence importante entre le fichier ssh\_config et le fichier sshd\_config** est que la plupart des directives du fichier ssh\_config sont des **sous-direcitves de la directive Host**.   
La **directive Host** permet de **spécifier différentes règles pour différents serveurs SSH** auxquels on se connecte.   
Par exemple, les éléments suivants appliqueraient la valeur 0 de **ConnectTimeout** lors de la connexion à srv01.profsavard.com et la valeur 600 lors de la connexion à test.example.org …

**Host srv01.profsavard.info  
 ConnectTimeout 0  
Host test.tux.org  
 ConnectTimeout 600**

**Remarque** …  
La **plupart des** **directives du fichier ssh\_config sont liées aux directives du fichier de configuration du serveur SSH**.

### Commande ssh

La **première fois que l’on essaie de se connecter** à un hôte à l’aide de la commande ssh, on est **invité à vérifier l'empreinte de la clé RSA** …  
**>> ssh** [**tux@srv01.monserveur.info**](mailto:tux@srv01.monserveur.info) **The authenticity of host 'srv01.monserveur.info (192.168.21.100)' can't be established.  
ECDSA key fingerprint is SHA256:VYhBBhvTXSwfT84uso27yc2kf5FV8iaqzPv8uwA+dlo.  
Are you sure you want to continue connecting (yes/no)? yes**

### Connexion à un serveur SSH

Avec le serveur openSSH en cours d'exécution, on **peut s’y connecter à l'aide du programme client ssh**.   
Afin de se connecter au serveur distant en utilisant le même compte d'utilisateur que l‘on utilise sur le client, on spécifie simplement le nom d'hôte ou l'adresse IP du serveur distant …

Depuis l’hôte client …  
**>> ssh <Identifiant>@<Nom DNS ou Adresse IP du serveur>  
>> ssh tux@srv01.profsavard.info  
>> ssh tux@192.168.1.100**

Si le **même identifiant est utilisé** sur le client et sur le serveur …  
**>> ssh <Nom DNS ou Adresse IP du serveur>  
>> ssh srv01.profsavard.info**   
**>> ssh 192.168.1.100**

Cette commande tente de se connecter à l'hôte distant 192.168.1.100 avec l'ID utilisateur tux.

* Si le compte d'utilisateur est défini pour l'**authentification par mot de passe**, on sera invité à entrer son mot de passe sur le système distant ;
* Si le compte d'utilisateur est configuré pour l'**authentification** **par certificat**, une invite de commande apparaîtra immédiatement sur le système distant.

Si on doit se **connecter sous un autre compte utilisateur**, on l’indique simplement sur la ligne de commande à l'aide du signe esperluette (@), comme une adresse électronique …  
**>> ssh -l <Nom d’utilisateur>@<Nom DNS ou Adresse IP du serveur>  
>> ssh -l kermit@srv01.profsavard.info**   
**>> ssh -l kermit@192.168.1.100**

Pour exécuter une commande sur le système distant, mais revenir immédiatement à son système client, on utilise la syntaxe …  
**>> ssh <Nom DNS ou Adresse IP du serveur> <Commande>  
>> ssh 192.168.1.100 "sudo apt update && sudo apt upgrade -y"**    
**>> ssh 192.168.1.100 "sudo systemctl status sshd.service"**

### Commandes scp et sftp

La commande scp utilise plusieurs des **mêmes options et fonctionnalités fournies par la commande ssh**.   
Pour copier un fichier de l’hôte actuel vers un hôte distant, on utilise la syntaxe suivante …  
**>> scp <Nom du fichier> <Nom DNS ou Adresse IP du serveur>:<Répertoire>  
>> scp /etc/hosts 192.168.1.22:/tmp**  
**>> scp monfichier.texte tux@192.168.1.77:/home/tux/Documents/monfichier.texte**

Pour transférer un fichier depuis l’hôte distant vers le système local, on utilise …  
**>> scp tux@192.168.1.77:/home/tux/Documents/monfichier.texte monfichier.texte**

Une fonctionnalité pratique de scp est la possibilité de copier des fichiers entre deux serveurs distants …  
**>> scp** tux**@192.168.1.100:monfichier.texte tux@192.168.1.101:monfichier.texte**

La commande sftp se connecte à un serveur SSH et fournit une interface client de type ftp.

**>> sftp** [**tux@srv01.profsavard.info**](mailto:tux@srv01.profsavard.info) **tux@srv01.profsavard.info's password:  
Connected to 192.168.1.13.  
sftp> pwd  
Remote working directory: /home/tux  
sftp> ls  
tux.texte  
sftp> get tux.texte .  
Fetching /home/tux/tux.texte to ./tux.texte  
sftp> ls  
tux.texte  
sftp> bye  
>>**

### 

### Connexion sans mot de passe

Si un utilisateur se connecte régulièrement à un serveur SSH spécifique, il peut vouloir **configurer des connexions sans mot de passe**.

Les étapes à suivre sont …

**Étape 1**Sur l’hôte client SSH, on exécute la commande ssh-keygen.   
Lorsque l’on est invité à entrer une phrase secrète, on appuie sur la touche Entrée pour laisser ce champ vide (ou on peut utiliser l'option -N avec la commande ssh-keygen, ce qui entraîne l'absence d'invite de phrase secrète) …  
**>> ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/tux/.ssh/id\_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/tux/.ssh/id\_rsa.  
Your public key has been saved in /home/tux/.ssh/id\_rsa.pub.  
The key fingerprint is:  
SHA256:cB/O6k0Wh/0KK/PycdoWhca0WIeWbLNxo6tIsTWVX98** [**root@srv01.profsavard.info**](mailto:root@srv01.profsavard.info) **The key's randomart image is:  
+---[RSA 2048]----+  
| . + |  
| & + .|  
| . . .O @ oo|  
| o.+++B o E|  
| S+\*ooo |  
| o. oo. |s  
| ...=.... |  
| .=+.B.. |  
| .\*\*.o |  
+----[SHA256]-----+**

Pour confirmer la présence des clés …  
**>> ls ./.ssh/  
id\_rsa id\_rsa.pub**

Pour afficher le contenu de la clé publique …  
**>> cat ./.ssh/id\_rsa.pub   
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDMbVkbxyTVo8ZCX4iJC3TBoEK2CNnlOax9LpmuQDq6iTK4FCGy1jovCjY7+/9/rvDeewVCDCn0mbvfOi/9P1mo7Pz86uBrhjwyhGsN+mGottcn+zxBOZS/6P6O/M/O71WOKNBWfDbxXdLxcCOhHvqx0f+hfGawGtgrF9xT42BvACWHqhqCiRIf4uzd69GapBP2w2FqEPVpauRjpkPH6BJNz2vhJf+jv4mJEsWsA52Zj5JOrVmSybjyryhr2ner4Pef6pd2/br1Ht2wHrmskZOF2SB3fdzylJvuyNnr0Fm+dQNVp9rc3hphMnWd5a6zakNB51yh28SxbURX9BF4ENb9 lsavard@debian-cli**

**Étape 2**Il existe maintenant une **clé publique dans le fichier ~/.ssh/id\_rsa.pub**.

Celui-ci **doit être copié sur le serveur distant dans le fichier ~/.ssh/registered\_keys** de son **compte sur l’hôte distant**.  
On devra peut-être créer ce répertoire et définir les autorisations sur 770).

Cela peut être fait avec les commandes suivantes …  
**>> ssh-copy-id -i id\_rsa.pub srv01.profsavard.info**

La clé publique va se retrouver dans le **fichier /home/lsavard/.ssh/authorized\_keys** sur le serveur.

**Étape 3**On définit les **autorisations du fichier allowed\_keys sur 640** …  
**>> ssh -l tux srv01.profsavard.info 'chmod 640 /home/tux/.ssh/authorized\_keys'  
tux@ srv01.profsavard.info 's password:**

Il est **maintenant possible de se connecter au serveur SSH sans mot de passe** …  
**>> ssh -l tux srv01.profsavard.info 'date'  
Linux debian-cli 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86\_64  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/\*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Mon May 24 15:05:36 2021 from 192.168.1.104**

Certains experts en sécurité soulignent que **le fait de ne pas utiliser de phrase secrète pour sa clé présente un risque** pour la sécurité. Cependant, **si on a une phrase secrète**, **chaque fois que l’on essaie de se connecter** au serveur SSH à partir du client, au lieu de demander le mot de passe, **on sera invité à entrer la phrase secrète** (ce que l’on tentait d'éviter en premier lieu) …

**Étape 1**on **démarre un nouveau shell bash avec l'utilitaire ssh-agent** …  
**>> ssh-agent /bin/bash**

**Étape 2**On **exécute la commande suivante** …  
**>> ssh-add ~/.ssh/id\_rsa**

À partir de ce moment, lorsque l’on **utilise le shell ssh-agent** pour se connecter à distance au serveur SSH, la **connexion a lieu sans mot de passe**.

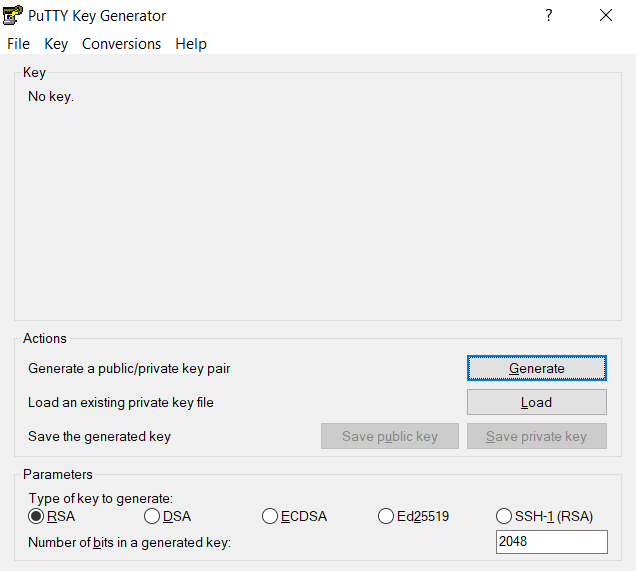
## Gestion des clés à l’aide de Putty

### Création une clé SSH à l’aide de PuTTY (sous Windows)

PuTTY est un client SSH répandu pour Windows.   
On peut l’utiliser pour se connecter à distance à un serveur Linux.   
Son logiciel compagnon, **Putty Key Genera tor** ou **PuTTYgen**, peut être **utilisé pour créer des clés SSH**.

Tout d’abord, si ce n’est déjà fait, il faut **télécharger le logiciel PuTTYgen**, qui **servira à générer la clé**.

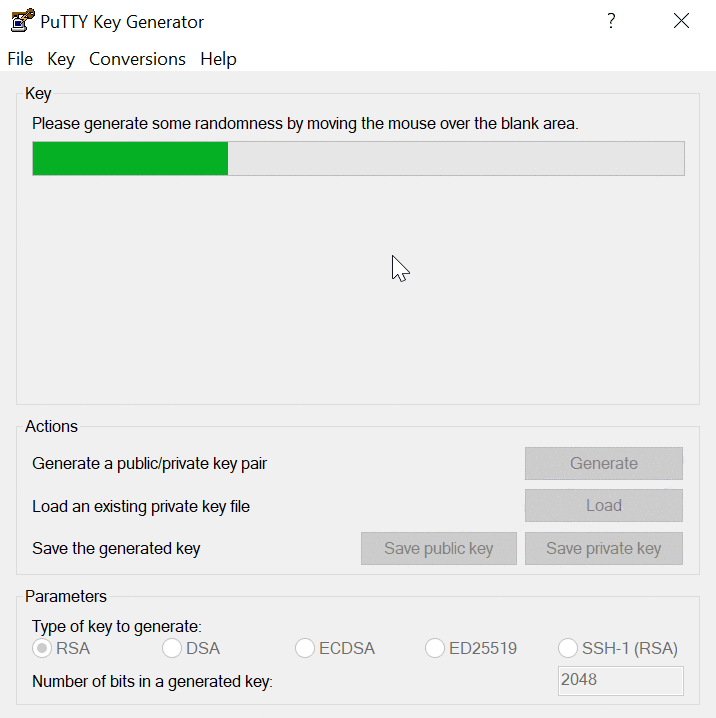
On exécute ensuite l’utilitaire.



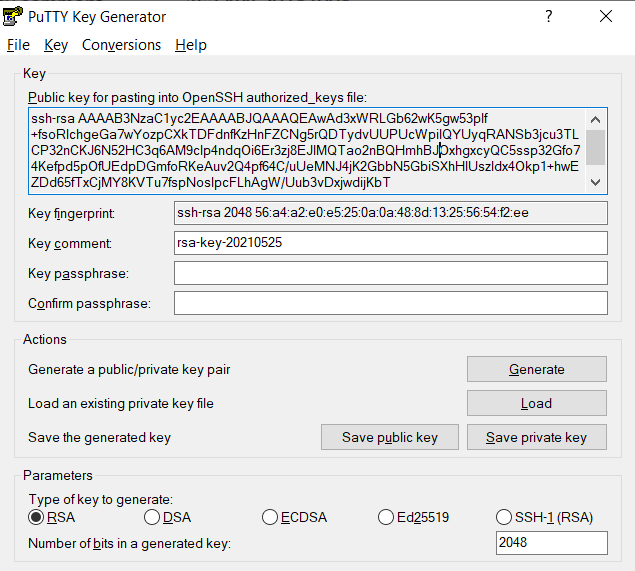
Dans un premier temps, on choisit le **type de clé.**   
Dans cet exemple, il s’agit d’une **clé RSA de 2048 bits**.

On clique sur **Générer** pour **lancer le processus de création**.

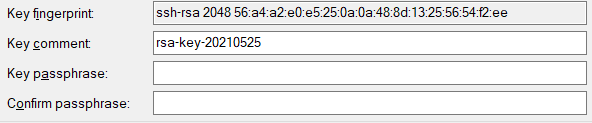
On **déplace le curseur de la souris** de façon aléatoire dans la zone située sous la barre de progression **afin d’accélérer la création** de la clé.



**PuttyGen**, à la **fin du processus de création**, affiche ce qui suit …



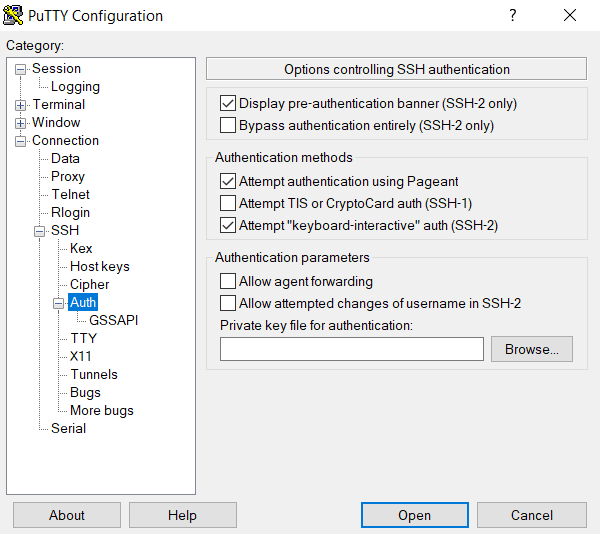
Bien que **cela n'est pas obligatoire** on peut saisir, par 2 fois, un **mot de passe**, qu'il faudra **noter ou retenir**.  
Ainsi celui qui arrive à copier cette clé privée ne pourra pas s'en servir sans saisir ce mot de passe.



Il faut enfin sauvegarder les clés publique et privée.

### Utilisation de la clé publique avec Putty

Afin d’utiliser sa clé pour **ouvrir une session SSH avec Putty** …  
**Connection > SSH > Auth** et on indique le **chemin sous la rubrique Private Key file for authentification**.



# En vrac

## Taux-limite les connexions

Il est possible de limiter la vitesse à laquelle une adresse IP peut établir de nouvelles connexions SSH en configurant le pare-feu. Si une adresse IP tente de se connecter plus de 10 fois en 30 secondes, toutes les tentatives suivantes échoueront, car les connexions seront rejetées.

Dans un environnement mono-utilisateur, pour modifier le nombre total de connexions en connexion simultanées en attente (pas encore autorisées) au système, il faudra définir dans sshd\_config …  
**MaxStartups 2:30:10**

Cet exemple autorisera deux connexions en attente.   
Entre la troisième et la dixième connexion, le système commence à supprimer de manière aléatoire les connexions de 30% à 100% lors de la dixième connexion simultanée.

Dans un environnement multi-utilisateur ou serveur, ces chiffres doivent être nettement plus élevés en fonction des ressources et de la demande en matière d'atténuation des attaques par déni d'accès. Définir un temps de connexion plus court (temps nécessaire pour conserver les connexions en attente en attente d'une autorisation) peut être une bonne idée car il libère les connexions en attente plus rapidement, mais au détriment de la commodité.  
**LoginGraceTime 30**

## Plus d'informations vers les journaux (*logs*)

Par défaut, le serveur openSSH se connecte à la fonction AUTH de *syslog*, au niveau INFO.   
Si on souhaite enregistrer davantage d'informations, telles que les tentatives de connexion infructueuses, il faudra augmenter le niveau de journalisation à VERBOSE.

Il est recommandé de consigner plus d'informations si vous êtes intéressé par le trafic SSH malveillant.  
Pour augmenter le niveau, identifier la ligne suivante dans le ***sshd\_config*** …  
**LogLevel INFO**  
et modifier pour …  
**LogLevel VERBOSE**

Désormais, tous les détails des tentatives de connexion SSH seront enregistrés dans le fichier /var/log/auth.log.

Si on utilise un autre port que celi par défaut (TCP 22) ou si on pense que son serveur est suffisamment caché pour ne pas à avoir besoin de beaucoup de sécurité, il est possible d’augmenter son niveau de journalisation et examiner son fichier auth.log de temps à autre.   
Si on rencontre un nombre important de tentatives de connexion frauduleuses, l’hôte est attaqué et on aura besoin de plus de sécurité.

Quelles que soient les précautions de sécurité qui ont été prises, il est possible de définir le niveau d’enregistrement sur VERBOSE pour une semaine et connaître le trafic parasite généré.  
Ainsi, il sera possible de constater à quel point l’hôte est attaqué.

## Affichage d’une bannière

Il peut être intéressant, du point de vue juridique, d’afficher un message de mise en garde lors de la connexion à un hôte depuis SSH.

Pour ajouter une bannière qui sera affichée avant l'authentification, identifier cette ligne …  
**#Banner /etc/issue.net**  
et le remplacer par …  
**Bannière /etc/issue.net**

Cela affichera le contenu du fichier ***/etc/issue.net***, qu’il est possible d’éditer selon ses besoins.   
Si on souhaite afficher la même bannière pour les utilisateurs SSH et pour les utilisateurs se connectant à une console locale …  
Banner /etc/issue

Exemple de bannière …

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**ALERT!   
You are entering into a secured area!   
Your IP, Login Time, Username has been noted and has been sent to the server administrator.  
This service is restricted to authorized users only. All activities on this system are logged.  
Unauthorized access will be fully investigated and reported to the appropriate law enforcement agencies.**

**ALERTE!   
Vous entrez dans une zone sécurisée!   
Votre adresse IP, heure de connexion, nom d'utilisateur a été noté et a été envoyé à l'administrateur du serveur.  
Ce service est réservé aux utilisateurs autorisés. Toutes les activités sur ce système sont enregistrées.  
Les accès non autorisés feront l’objet d’une enquête approfondie et seront signalés aux services de détection et de répression appropriés.**

**¡ALERTA!   
Usted está entrando en un área segura!   
Su IP, tiempo de inicio de sesión, nombre de usuario se anotó y se envió al administrador del servidor.  
Este servicio está restringido a usuarios autorizados solamente.   
Todas las actividades en este sistema están registradas.  
El acceso no autorizado se investigará por completo y se informará a las agencias de aplicación de la ley correspondientes.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Infos pratiques côté Client**~/.ssh/config : Configuration simple

Il existe un outil qui simplifie la gestion : **les alias SSH**.  
Si on a l'habitude de se connecter à plusieurs serveurs SSH, (noms d'hôtes différents, ports différents) et que l'on veut simplifier la saisie de commande, il est possible de créer des alias …  
**>> ssh -p 5222 tux@rv01.profsavard.info**

Il suffit de créer le **fichier ~/.ssh/config** et d'y ajouter les lignes suivantes ...  
**Host srv01.profsavard.info  
 HostName 192.168.10.100  
 Port 5222  
 User tux**



Pour se connecter au serveur srv01.profsavard.info, il suffit de saisir la commande ...  
**>> ssh srv01**