SSH : Secure SHell

De l'utilisateur à l'administrateur

Version novembre 2008

F. Bongat [fbongat@lmd.ens.fr]

Présentation

- Sécuriser des connexions à distance : *Secure Shell*
  - SSH permet de sécuriser les communications des réseaux
    - Utilise pour cela de la cryptographie
    - SSH est composé d'un ensemble d'outils permettant des connexions sécurisées entre des machines. Ces outils ont pour but de remplacer les utilitaires de connexions classiques n'utilisant pas de chiffrage.
      - Remplace : rcp, rlogin, rsh, telnet (ftp par sftp en SSH V2)
    - SSH chiffre et compresse un canal de communication qui sécurise les données transmises (permet d'éviter les sniffers réseaux)
    - Non seulement le mot de passe est chiffré lors de la connexion mais les informations circulant sur le réseau entre les deux machines le sont aussi.

Présentation

- Terminologie de SSH :
  - **C'est un protocole**
    - c'est-à-dire c'est une méthode standard permettant à des machines d'établir une communication sécurisée
    - décliné en 2 versions :
      Version 1 et ***version 2*** : le protocole v1 possédait une faille permettant à un pirate d'insérer des données dans le flux chiffré
  - **C'est aussi un produit** :
    - SSH Communications Security Inc (V1 et V2)
      Initialement développé par Tatu Ylönen (payant)
      dernière version gratuite v1.2.12
    - OpenSSH du projet OpenBSD (V1 et V2)
      apparaît en 1999, aujourd'hui c'est le plus utilisé
      Produit en accord avec la législation française sur la cryptographie
      http://www.ssi.gouv.fr/fr/reglementation/index.html#produits_crypto
  - **Ce sont aussi des commandes en ligne**

Présentation

- OpenSSH
  - Site : http://www.openssh.org
  - Version logicielle actuelle : openssh-5.1.tar.gz
    - Existe en format packagé pour toutes distributions
  - Subit un audit permanent du code
  - OpenSSH utilise principalement :
    - OpenSSL
    - Zlib pour la compression des flux
    - Perl lors de l'installation, pour des applications tiers (ssh-copy-id…)
    - Ainsi que des générateurs d'entropie
      - PRNGD (Pseudo Random Number Generator Daemon)
      - /dev/random

Présentation

- La boîte à outils SSH est généralement composée de :
  - Serveur : *sshd*
  - Clients : *ssh*, *scp*, *sftp* (ssh = slogin)
  - Des outils de gestion: *ssh-add*, *ssh-agent*, *ssh-keygen*, *ssh-keyscan*
  - Les fichiers de configuration (OpenSSH) sont souvent dans:
    - Pour le serveur : /etc/ssh
    - Pour les clients : /etc/ssh et $HOME/.ssh
- Fonctionnement sur le schéma d'un système client – serveur

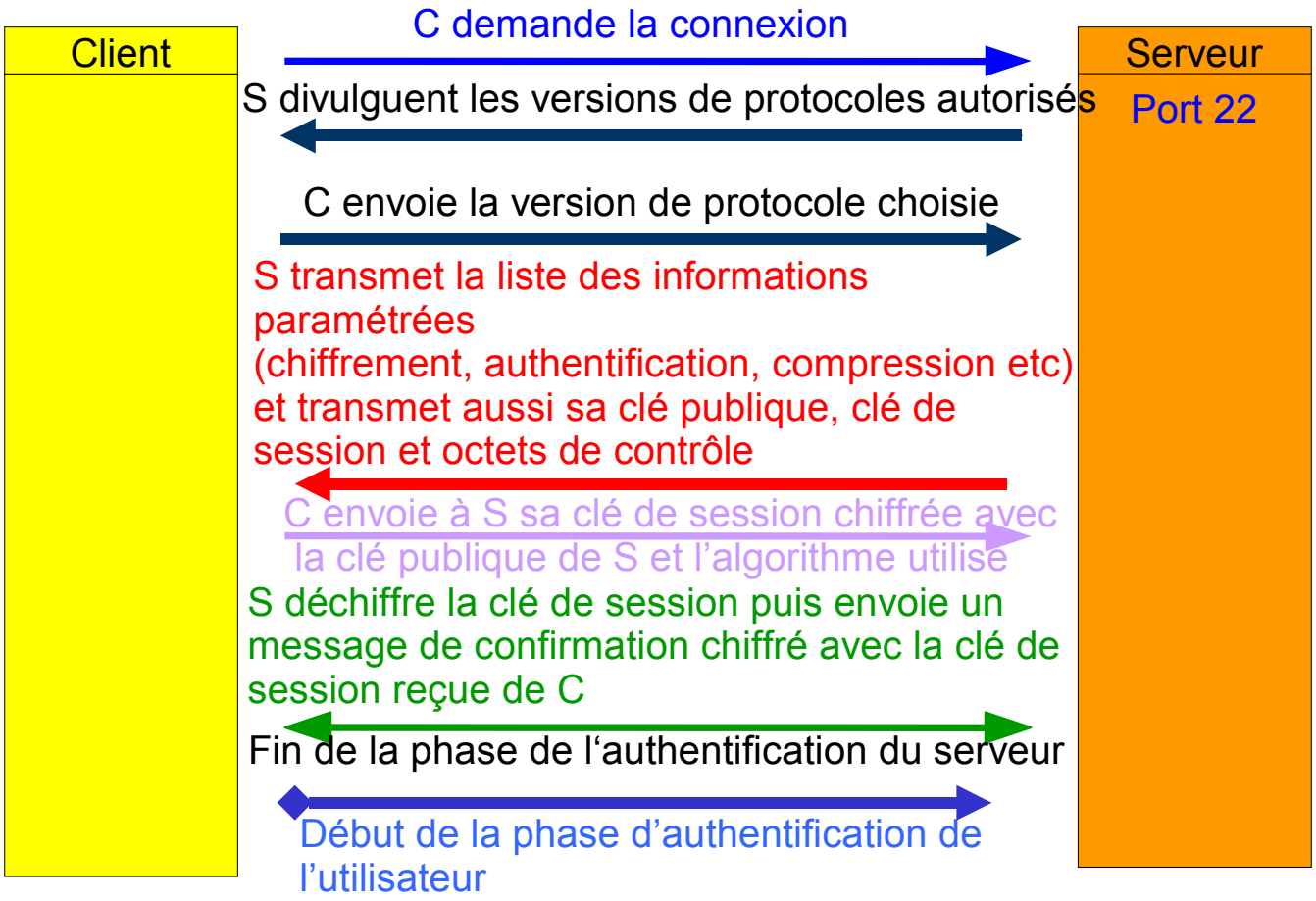
  Les clients *ssh* demandent une ouverture de connexion au serveur *sshd*

Présentation

- Cryptographie dans SSH
  - Repose sur les algorithmes d'OpenSSL
    - Algorithmes asymétriques:
      - RSA et DSA
    - Algorithmes symétriques:
      - 3DES, Blowfish, AES, Arcfour ...
- Linux/Unix : PAM et SSH
  - Gestion centralisée des services:
    - Possibilité d'activer le support de l'interface PAM(Pluggable Authentication Modules) au niveau de sshd

SSH

- **Fonctionnement du protocole SSH**
  - Mise en place d'un canal sécurisé



Présentation

- L'authentification des serveurs
  - Le principe d'authentification du serveur se fait par chiffrement à clé publique du protocole SSH.
  - Le client doit donc connaître la clé publique du serveur sur lequel il veut se connecter avant toute connexion. Ainsi, il existe un mécanisme pour la machine cliente pour stocker les clés d'un serveur afin de les réutiliser ensuite;
  - Il pourra ainsi vérifier la clé d'un serveur à chaque nouvelle connexion avec celle enregistrée lors de la première connexion;
  - Cela permet d'éviter les attaques du type *man-in-the-middle*

Présentation

- L'authentification des utilisateurs
  - Une fois que la connexion sécurisée est mise en place entre le client et le serveur, le client doit s'identifier sur le serveur afin d'obtenir un droit d'accès.
    - **Par mot de passe**: Le client envoie un nom d'utilisateur et un mot de passe au serveur au travers de la communication sécurisé et le serveur vérifie si l'utilisateur concerné a accès à la machine et si le mot de passe fourni est valide
    - **Par clés publiques** : Si l'authentification par clé est choisie par le client, le serveur va créer un *challenge* et donner un accès au client si ce dernier parvient à déchiffrer le challenge avec sa clé privée
    - **Par hôte de confiance** : système équivalent aux systèmes utilisant rhost ou hosts.equiv en utilisant les clés publiques des serveurs
    - **Et aussi par Kerberos, SmartCard, PAM**

Présentation

- VPN (virtual Private Network) avec SSH
  - Le support d'un protocole de VPN est une fonctionnalité apparue récemment dans OpenSSH.
  - Le principe est le même que pour OpenVPN mais il est basé sur le protocole SSH (différents des tunnels TCP SSH)
  - Supporte les interfaces réseaux virtuelles Tun/Tap (niveau 3/niveau 2)
  - Option à activer du côté serveur : *PermitTunnel*

Présentation

- Séparation de privilèges du serveur SSH (chroot)
  - Chaque processus lors de son exécution ne possède que les privilèges nécessaires, et n'a alors accès qu'aux éléments nécessaires à son exécution
    - cette mise en place augmente la sécurité des applications
  - *chroot* est une commande des systèmes d'exploitation UNIX permettant de changer pour un programme le répertoire racine de la machine hôte
  - Cette commande permet d'isoler l'exécution d'un programme et d'éviter ainsi certaines malveillances
    - exemple: l'exploitation d'un dépassement de tampon, pour ensuite accéder au répertoire racine de la machine hôte.

Présentation

- Séparation de privilèges du serveur SSH (chroot)
  - sshd utilise deux processus :
    - le processus père privilégié contrôle les progrès du processus fils non privilégié (effectue le minimum nécessaire : la réussite de l'authentification est déterminée par le processus père)
    - Le processus fils est non privilégié. Ceci est atteint en changeant ses uid/gid vers un utilisateur non privilégié

```
root      4933  3423  0 14:30 ?        00:00:00 sshd: fbongat [priv]
fbongat   4935  4933  0 14:30 ?        00:00:00 sshd: fbongat@pts/0
```

Présentation

- Mise en cage : chroot
  - OpenSSH intègre la directive *ChrootDirectory* permettant de chrooter dans un répertoire après la connexion.
  - Cette fonctionnalité est principalement utilisée pour du SFTP
  - on peu parfaitement configurer l'ensemble pour les utilisateurs SSH
  - Il faut construire l'environnement nécessaire

Présentation

- Clients/serveurs multi plates-formes :
  - Windows (clients gratuits) :
    - SSH Tectia
      - SSH Secure Shell for Workstation (nom du produit)
      - http://www.ssh.com/support/downloads/secureshellwks/non-commercial.html
      - Seul le client Tectia 3 (ex-ssh.com) est gratuit, les versions Tectia sont payantes. Inconvénient de la version gratuite : le module de connexion à base de certificats X509 est désactivé ainsi que le transfert d'agent
    - Putty
      - Clients regroupant toutes les commandes connues sous OpenSSH
      - http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html
    - Winscp (v3)
      - Outil graphique de transfert de fichiers (scp/sftp, donc pas de ssh) très performant
      - http://winscp.net/eng/docs/lang:fr/
    - Serveur gratuit pour windows via Cygwin
      - http://www.cygwin.com/
      - **sshd** aussi disponible seul sans l'environnement Cygwin complet (attention au problèmes de sécurité liés à ce serveur moins robuste)

Présentation

- Clients/serveurs multi plates-formes :
  - Mac
    - MacOS 9 et inférieur :
      - client nifty-telnet (v1 uniquement)
      - http://www.lysator.liu.se/~jonasw/freeware/niftyssh/
    - MacOS X :
      - clients/serveur natifs (OpenSSH)
      - fugu : outil graphique de transfert de fichiers
        - http://www.columbia.edu/acis/software/fugu/
  - Unix (tous)
    - OpenSSH
      - http://www.openssh.org/
      - Proposé en standard dans la plupart des distributions Unix
    - Ssh.com (Tectia): existe aussi en version Unix

Côté client

- SSH vu du côté client …

Côté client : unix - ssh

- Utilisation simple : **ssh**
  - Connexion distante (alternative à telnet, rlogin) :
    - Syntaxe : **ssh** *login*@machine_distante

```
[root@spirou root]#
[root@spirou root]# ssh fbongat@spip
fbongat@spip's password:*******
[fbongat@spip fbongat]$
[fbongat@spip fbongat]$
```

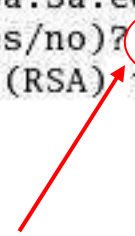
    - Ou bien avec l'option ***-l***

```
[root@spirou root]#
[root@spirou root]# ssh -l fbongat spip
fbongat@spip's password:
[fbongat@spip fbongat]$ *******
[fbongat@spip fbongat]$ _
```

Côté client : unix - ssh

- Utilisation simple : **ssh**
  - Première connexion distante :
    - message d'alerte lors d'une connexion vers une nouvelle machine distante

```
[fbongat@spirou fbongat]$ ssh fbongat@spip
The authenticity of host 'spip (172.16.158.20)' can't be established.
RSA key fingerprint is 04:64:a8:06:a0:b9:83:4a:0a:3a:ed:bf:bc:a2:e2:7c.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'spip,172.16.158.20' (RSA) to the list of known hosts
.
fbongat@spip's password: *******
[fbongat@spip fbongat]$ _
```

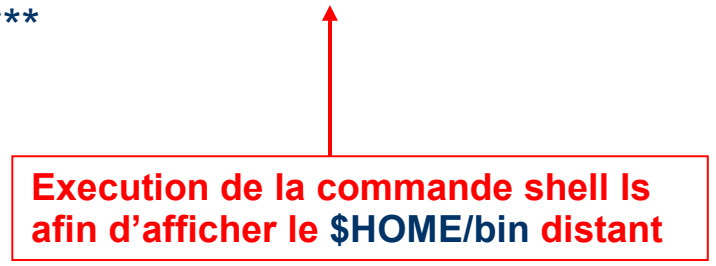


    - Il est nécessaire de répondre « ***yes*** » à la question demandée pour se connecter. En fait, il s'agit d'autoriser l'enregistrement de la clé publique du serveur distant dans un fichier de configuration (known_hosts)

Côté client : unix – ssh : rsh

- Utilisation simple : **ssh**
  - Connexion distante (alternative à rsh) :
    - Utilisation d'une commande shell à distance
      - Ici on liste le contenu du répertoire $HOME/bin sur la machine distante fantasio

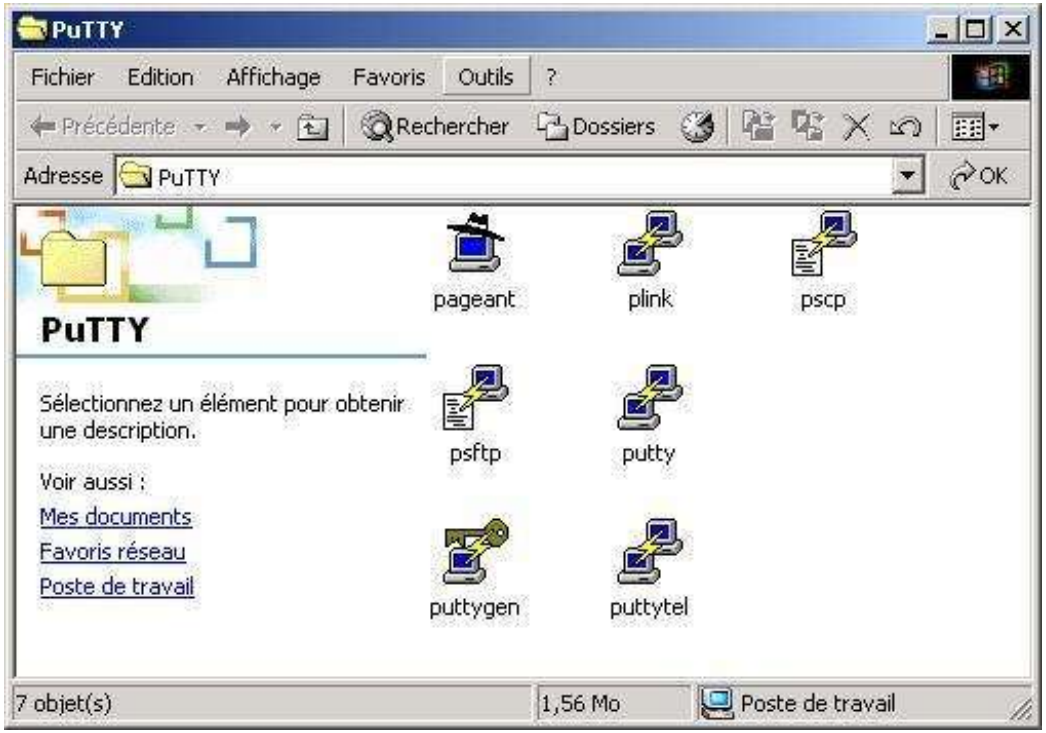
```
[root@spirou root]# ssh fbongat@fantasio ls bin/
fbongat@fantasio's password: *******
du.csh
iptables.conf
quota.bash
script.csh
test.csh
[root@spirou root]#
[root@spirou root]#
```



Côté client : PuTTY - ssh

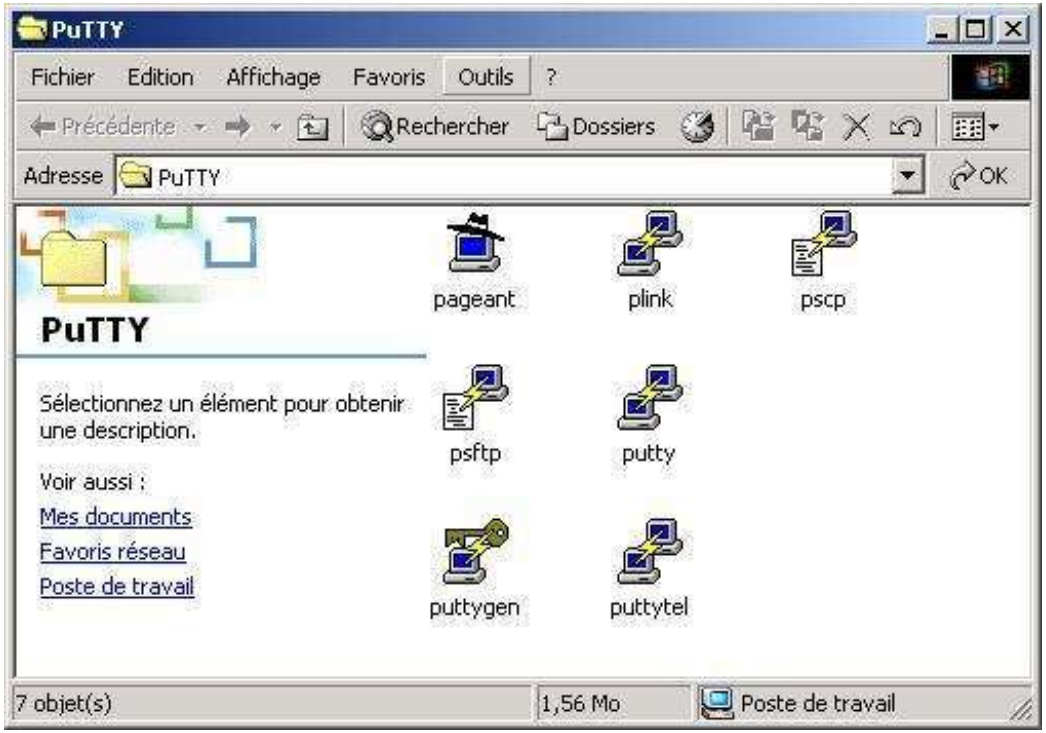
- SSH et Windows : PuTTY
  - Implémentation libre
  - Proche d'OpenSSH
    - Boîte à outils qui comprend un ssh, sftp, scp, ssh-agent et utilise des clés (compatibles avec les clés OpenSSH)
    - 7 binaires (ou un fichier zip) dont 5 indispensables (pageant, pscp, psftp,putty et enfin puttygen) à copier (ou décompresser) dans le dossier :

      C:\Program Files\PuTTY



Côté client : PuTTY - ssh

- SSH et Windows : PuTTY
  - ***pageant*** : agent d'authentification (*voir chapitre authentification forte*)
  - ***plink*** : ssh en mode commande dans une console (~ Invite de Commande)
  - **pscp** : scp en mode console
  - ***psftp*** : sftp en mode console
  - ***putty*** : ssh en mode graphique
  - ***puttygen*** : gestion des clés en mode graphique
  - ***puttytel*** : telnet en mode graphique (pas besoin !)

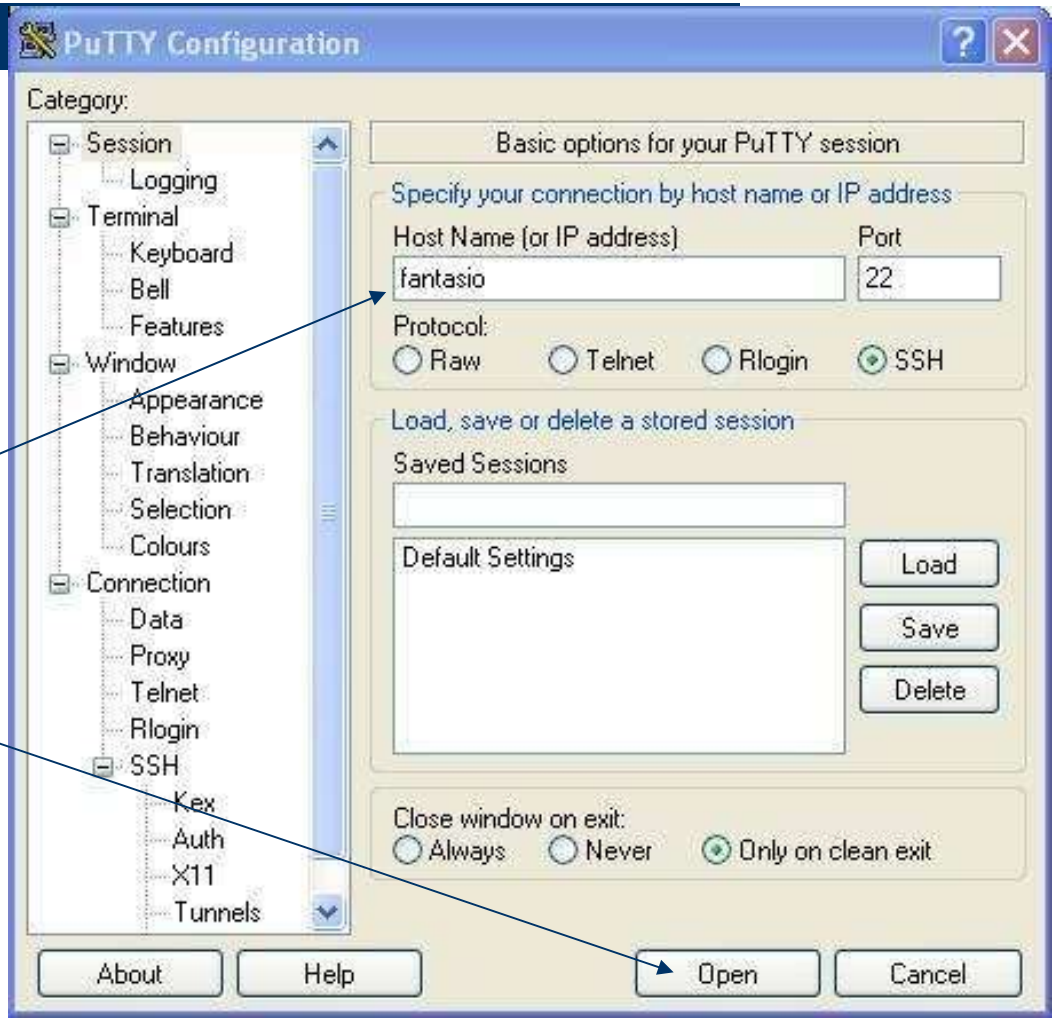


Côté client : PuTTY - ssh

- SSH et Windows : PuTTY
  - SSH
    - connexion rapide avec putty
      - Double cliquer sur l'icône PuTTY
      - Puis remplir le champ suivant:

        Host Name

        Et valider « ***open*** » pour lancer la connexion



Côté client : PuTTY - ssh

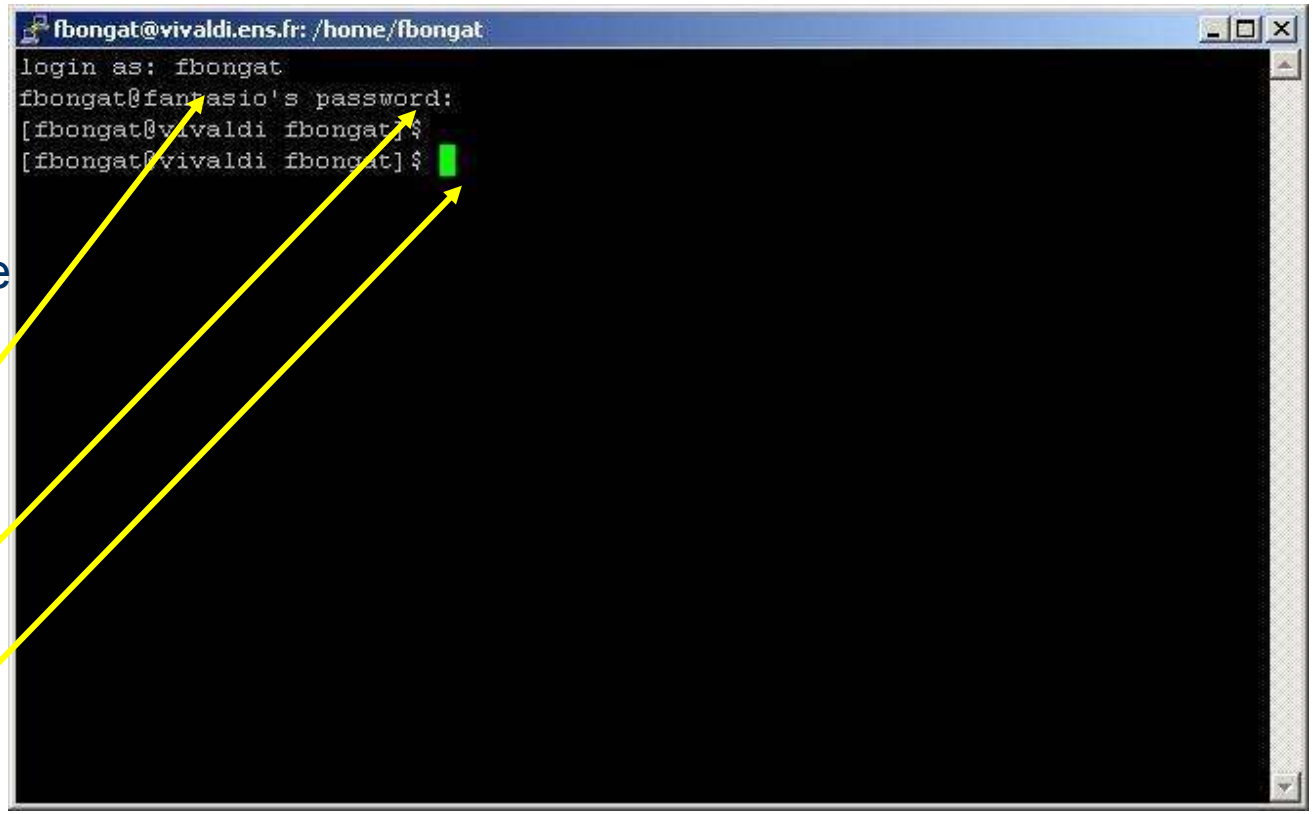
- SSH et Windows : PuTTY
  - SSH
    - connexion rapide avec putty

Ajout de la clé publique dans la base de registres (équivalent au fichier known_hosts)

Répondre « oui » pour passer à la suite !

Côté client : PuTTY - ssh

- SSH et Windows : PuTTY
  - SSH
    - connexion rapide avec putty



Taper le login après le prompt « *login as* »

Puis entrer le mot de passe

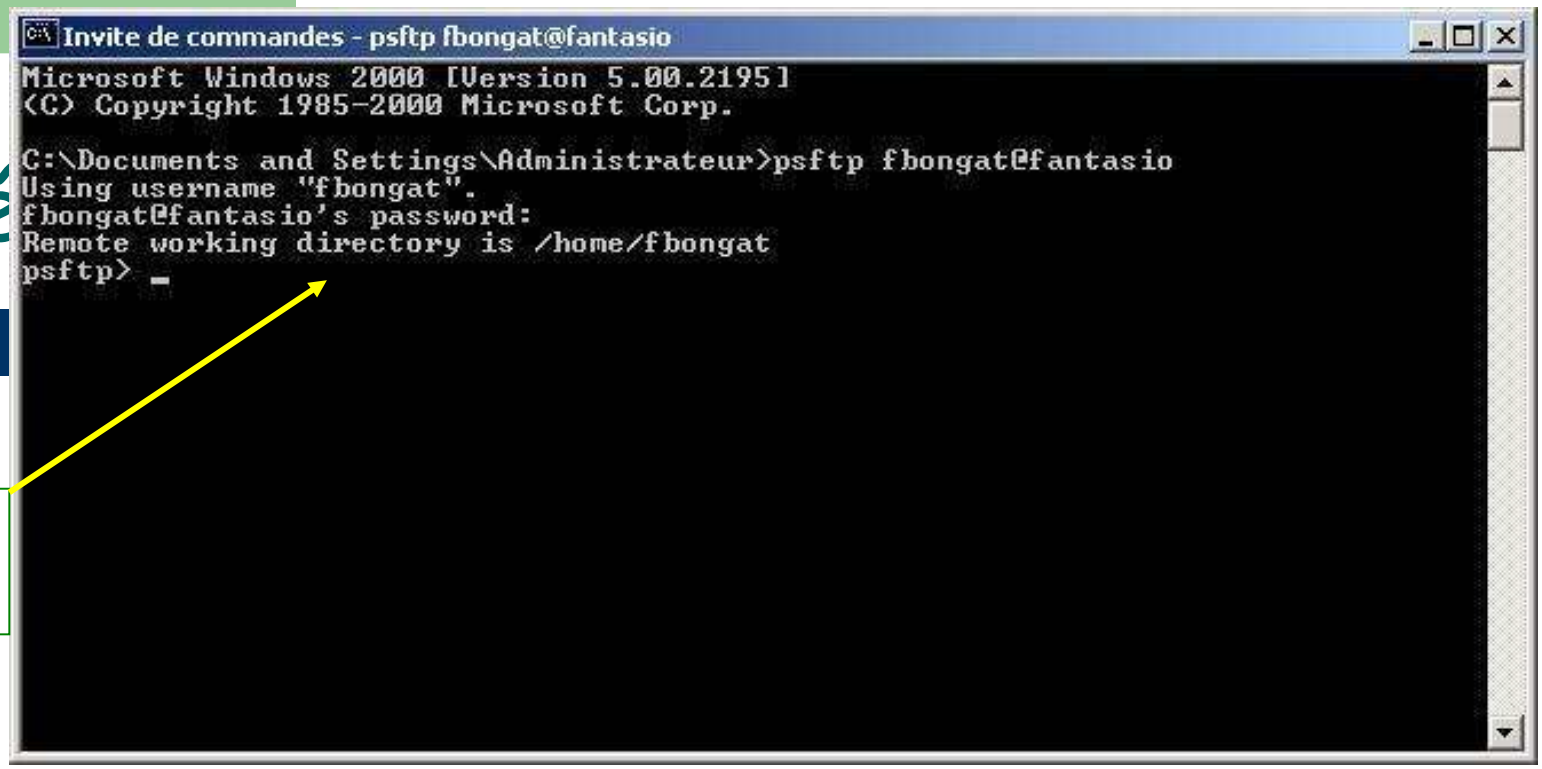
Connexion en ssh sur la machine distante avec PuTTY

Côté client : unix - sftp

- Utilisation simple : **sftp**
  - Transfert de fichiers (une alternative sécurisée à ftp)
    - **sftp login@machine** uniquement, pas d'option –l
    - Les commandes sont les mêmes qu'avec ftp (put, get, mput etc…)

```
[root@spirou root]# sftp fbongat@spip
Connecting to spip...
fbongat@spip's password:
sftp>
sftp> dir
.
..
.Xauthority
.bash_history
.bash_logout
.bash_profile
.bashrc
.emacs
.gtkrc
.ssh
.viminfo
sftp>
```

Côté



psftp fonctionne comme sftp dans un shell

- SSH et Windows : PuTTY
  - SFTP
    - Ouvrir une console « ***Invite de commandes*** »
    - Tapper ***psftp*** dans cette fenêtre

      psftp login@machine
    - **!** Si la commande n'est pas trouvée, référez vous à la partie configuration avancée pour configurer le PATH

Côté client : unix - scp

- Utilisation simple : **scp**
  - Transfert de fichiers (alternative sécurisée à rcp)
    - scp est une commande de copie de fichiers (ou répertoires) entre 2 machines à travers le réseau

```
[root@spirou temp]# ls
spirou.txt
[root@spirou temp]# scp spirou.txt fbongat@spip:tmp/
fbongat@spip's password:  *******
spirou.txt                                    100%   45    67.2KB/s   00:00
[root@spirou temp]#
[root@spirou temp]# scp fbongat@fantasio:bin/iptables.conf .
fbongat@fantasio's password: *******
iptables.conf                                 100% 1072     1.1MB/s   00:00
[root@spirou temp]#
```

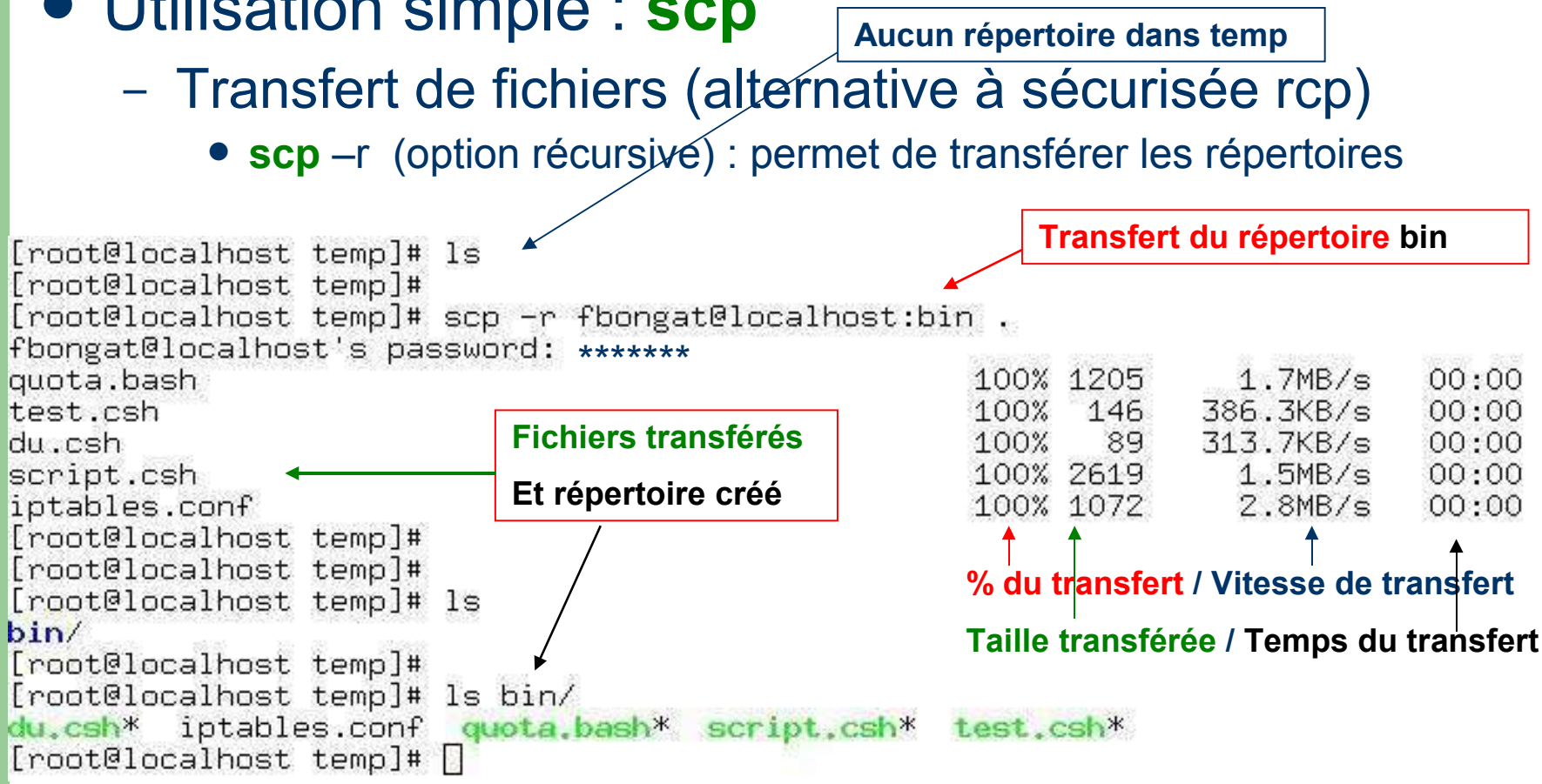
Envoi d'un fichier vers une machine

Récupération d'un fichier sur une machine distante

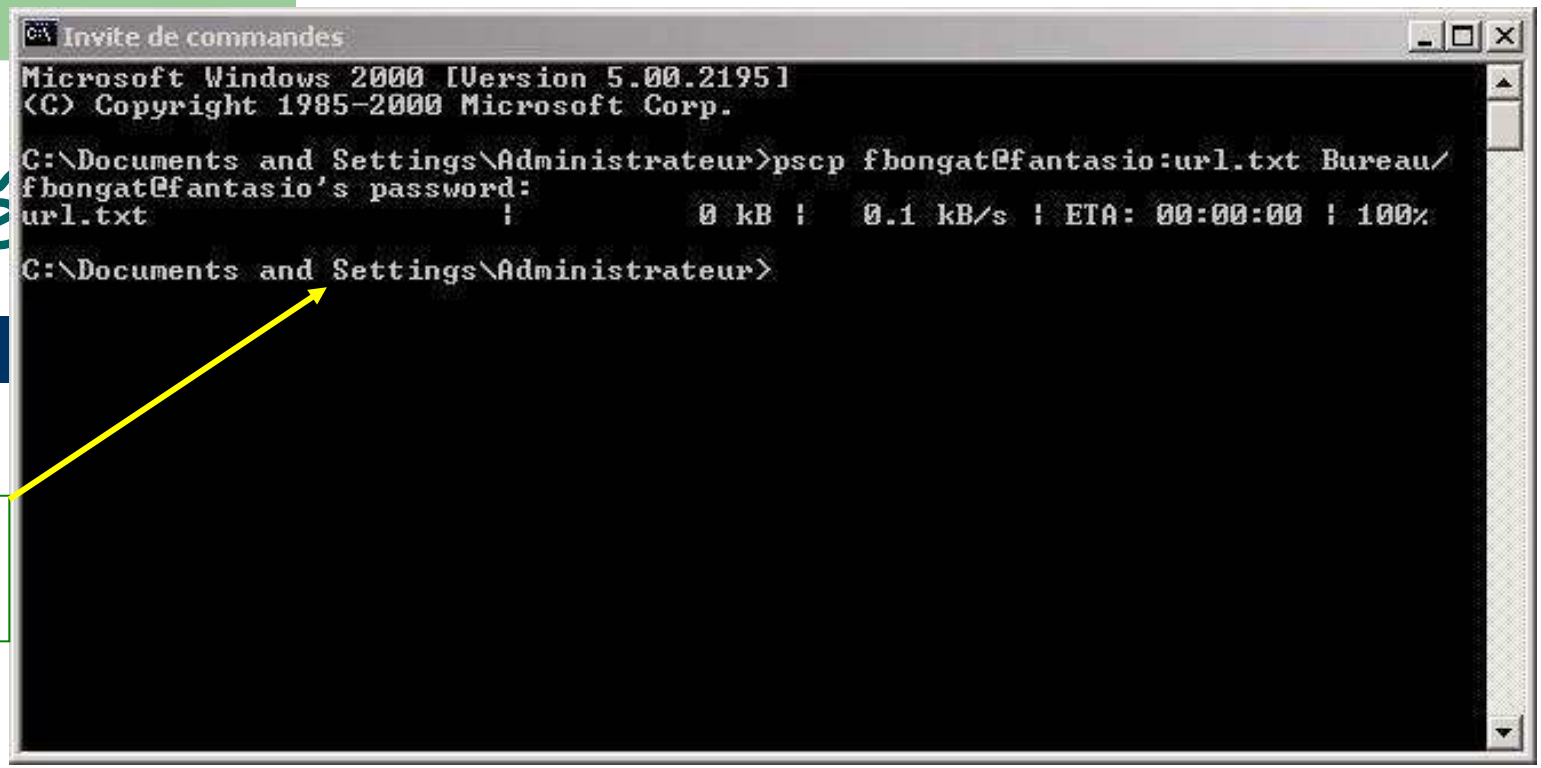
Informations sur le transfert

Côté client : unix - scp

- Utilisation simple : **scp**
  - Transfert de fichiers (alternative à sécurisée rcp)
    - **scp** –r (option récursive) : permet de transférer les répertoires



Côté



pscp fonctionne comme scp dans un shell

- SSH et Windows : PuTTY
  - SCP
    - Ouvrir une console « ***Invite de commandes*** »
    - Tapper ***pscp*** dans cette fenêtre

      pscp fichier **login@machine:**
    - **!** Si la commande n'est pas trouvée, référez vous à la partie configuration avancée pour configurer le PATH

Côté client : unix - fichiers impliqués

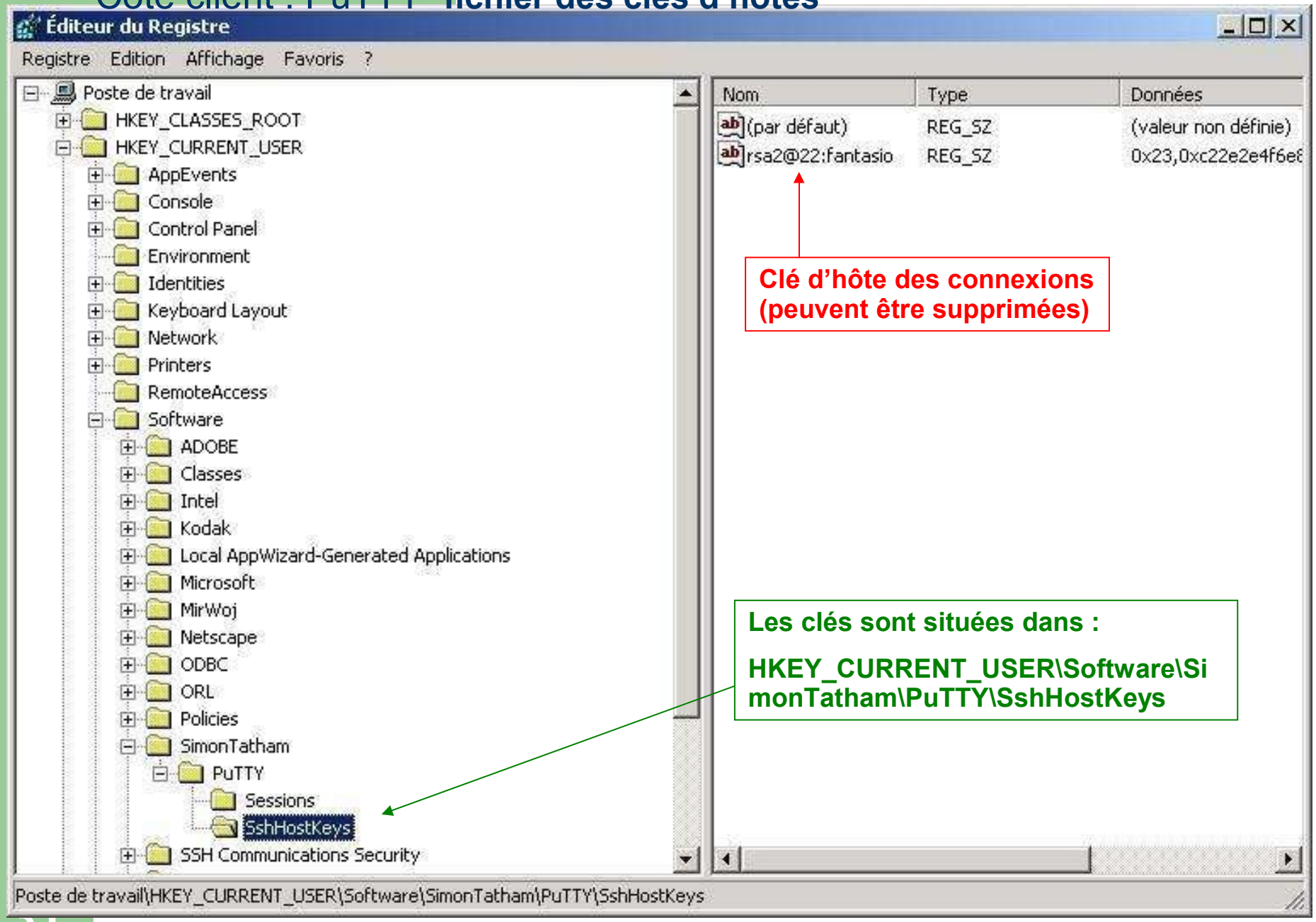
- Structure des fichiers impliqués du côté utilisateur :
  - Répertoire ssh par utilisateur : $HOME/.ssh
  - 2 fichiers :
    - known_hosts : contient les clés publiques des serveurs sur lesquels l'utilisateur s'est connecté (vérifie ainsi si un serveur n'a pas été substitué ou changé)
    - config : personnalisation des configurations clientes
      - Exemple : contenu du fichier ***config*** :

        Host fantas
        Hostname fantasio.bdnet
        User fbongat
      - lors d'une connexion ssh et que les comptes sur les 2 machines sont différents, on pourra grâce à la configuration ci-dessus faire:

        ssh fantas (à la place de ssh fbongat@fantasio.bdnet)

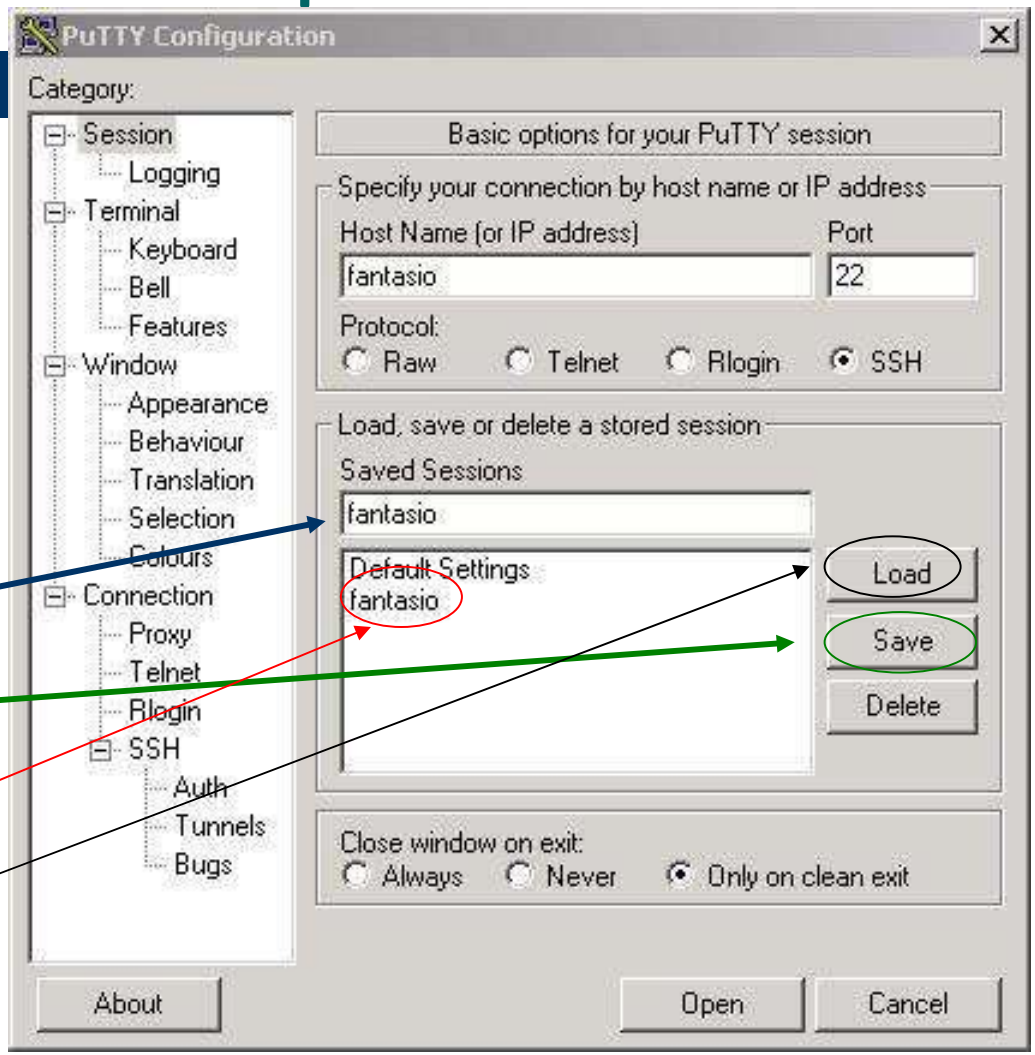
        qui renverra systématiquement le user **fbongat**, ce qui permettra de ne plus spécifier l'utilisateur avec les options **-l** ou **@**

Côté client : PuTTY fichier des clés d'hôtes



Côté client : PuTTY - profiles

- SSH et Windows : PuTTY
  - Configuration avancée
    - Rattacher les options sélectionnées à un profile d'utilisateur
      - Donner un nom au profile
      - Puis sauver : ***save***
    - Utiliser le profile
      - le sélectioner
      - Puis le charger ***load***



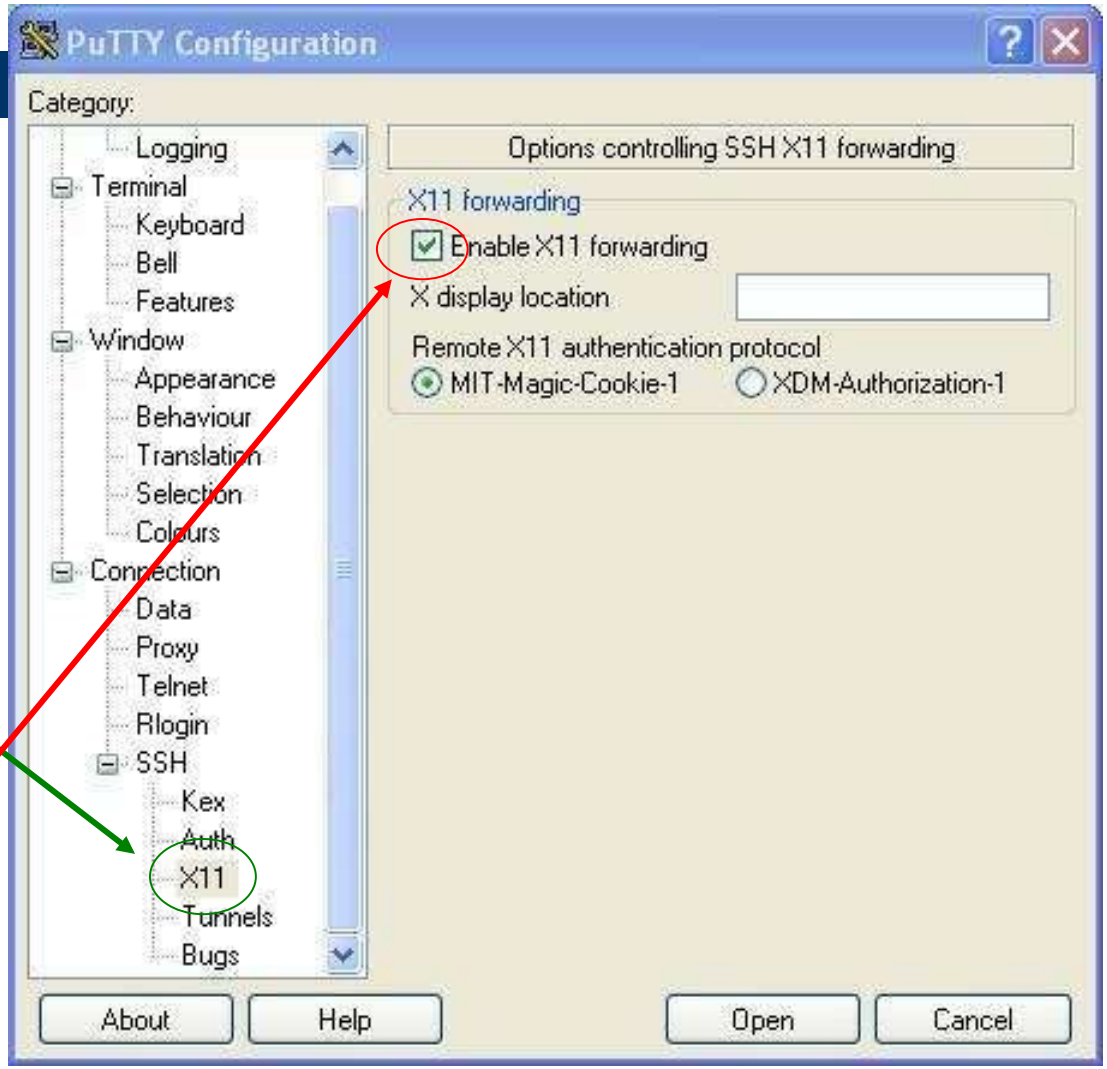
côté client : ssh et X11

- SSH et les applications graphiques Unix :
  - relaye simplement toutes applications X11 à travers le canal chiffré
    - Ne pas configurer de variable *$DISPLAY* dans les scripts de connexion (.cshrc, .profile, .bashrc etc..), ssh doit remplir lui-même cette valeur
      - *Donc c'est plus simple que telnet !*
  - Il est nécessaire d'avoir un serveur X11 sur la machine du client et de le mettre en fonctionnement
  - **Pour windows**, il existe un serveur X11 gratuit, simple et efficace qui interagit très bien avec PuTTY
    - ***Xming : http://freedesktop.org/wiki/Xming***
  - L'option ***-X*** active l'X11 en cas de non configuration de la variable ***ForwardX11*** dans le fichier ***/etc/ssh/ssh_config*** : voir partie *côté serveur*

    ssh –X login@machine

côté client : ssh et X11

- SSH et les applications graphiques Unix :
  - Avec le client Windows, il faut spécifier le demande de redirection du traffic X11 Unix dans la session ssh
  - ***Option : X11 Forwarding*** dans PuTTY :
    - Lancer ***putty***
    - Menu ***SSH*** → ***Tunnels***
    - Cocher la case : ***Enable X11 forwarding***
  - N'oubliez pas sous Windows, il faut aussi lancer un émulateur X11 (Xming)



Côté serveur

- SSH vu du côté serveur …

Côté serveur : sshd

- Le serveur : sshd
  - Répertoire du serveur : /etc/ssh
    - Les fichiers de configuration :
      - sshd_config , ssh_config, denyusers
    - Les fichiers des clés privées/publiques du serveur :
      - Clés compatible V1 (ssh v1/ssf)
        - ssh_host_key (privée) , ssh_host_key.pub (publique)
      - Clés compatibles V2 (ssh v2)
        - ssh_host_dsa_key (privée) , ssh_host_dsa_key.pub (publique)
        - ssh_host_rsa_key (privée) , ssh_host_rsa_key.pub (publique)

Côté serveur : configuration

- Configuration du serveur : sshd
  - Fichiers de configuration :
    - sshd_config : paramétrages lors des connexions vers le serveur local, ce fichier s'applique au démon sshd
    - ssh_config : paramétrages base et général du client ssh, ce fichier s'applique donc pour les commandes ssh, scp, sftp

Côté serveur : sshd_config

sshd_config

(partie 1)

- **Protocol :** choix du protocole à utiliser (initialiser de préférence à 2 ou si besoin d'une compatibilité SSH V1, laisser : 2,1)
- **PermitRootLogin** : autorise le compte *root* à se connecter (de préférence à initialiser à ***no***)
- **StrictModes yes** : vérifie les permissions des fichiers et répertoires importants (accès au propriétaire uniquement)
- **RSAAuthentication yes** : méthode d'authentification par RSA uniquement en V1 et V2
- **PubkeyAuthentication yes** : méthode d'authentification forte (rsa ou dsa) en V2 uniquement
- **AuthorizedKeysFile .ssh/authorized_keys** : nom et localisation du fichier de clés publiques individuelles sur les hôtes locaux.

Côté serveur : sshd_config

sshd_config

(partie 2)

- PasswordAuthentication yes : autorise la connexion par mot de passe
- PermitEmptyPasswords no : interdit les connexions sans mot de passe
- X11Forwarding yes : active le transfert X pour sshd
- X11DisplayOffset 10 : réservation d'un numéro d'affichage X11 afin d'éviter les collisions entre sshd et le vrai serveur X.
- X11UseLocalhost yes : bind sur l'interface de la boucle locale, permet d'éviter les problèmes de proxy
- Subsystem sftp /usr/lib/ssh/sftp-server : active le système de transfert de fichiers via sftp

Côté serveur : sshd_config

- Filtrage d'utilisateurs ou de groupes
  - Variables : *AllowUsers* et *DenyUsers*

    AllowGroups et *DenyGroups*
    - Permettre l'accès à certains utilisateurs et pas d'autres un accès ssh
      - Configuration pour autoriser les deux utilisateurs (fbongat et bob) et aucun autre à se connecter en ssh:

        DenyUsers
        AllowUsers *fbongat bob*

Côté serveur : ssh_config

ssh_config

- Host * : spécifie les hôtes concernés par la configuration qui suit (adresse ip ou nom DNS, * = toutes)
- ForwardAgent yes : indique à l'agent que l'agent d'authentification doit être renvoyé vers la machine distante
- ForwardX11 yes : autorise la redirection du serveur graphique (possibilité de lancer des applications graphiques dans la session ssh)
- ForwardX11Trusted yes: n'oblige pas le client ssh à créer un « untrusted X cookie » de sorte que les attaques sur la retransmission X11 ne puissent pas devenir des attaques des clients X11 depuis une machine distante

Côté serveur : ssh_config

ssh_config

- VerifyHostKeyDNS yes : Le client ssh peut vérifier la clef publique RSA/DSA d'un serveur SSH à l'aide du DNS

- StrictHostKeyChecking yes/ask/no
  - no : automatise la gestion des clés d'hôtes dans *known_hosts*. Si la clé n'existe pas, la connexion ne sera pas refusée, et sera rajoutée sans le demander à l'utilisateur (l'utilisateur ne verra pas les changements de clés du serveur)
  - ask : demande à l'utilisateur s'il veut ajouter la clé dans le fichier *known_hosts*, puis permet la connexion, si la clé du serveur change, un message d'avertissement sera envoyé (voir page 31) à l'utilisateur

Côté serveur : ssh_config

ssh_config

- StrictHostKeyChecking
  - yes : vérifie que la clé publique de l'hôte distant existe sur l'hôte qui cherche à se connecter et ensuite autorise la demande de connexion. Une non connaissance de la clé ou un changement de clé du serveur implique alors un échec de la connexion ssh
    - Il faut donc fournir la clé publique du serveur distant à la configuration cliente qui cherche à se connecter
    - */etc/ssh/ssh_known_hosts* : base de données des clés d'hôte
    - On ne peut donc pas se connecter la première fois sans avoir la clé installée

Côté serveur : StrictHostKeyChecking

Exemple d'une première connexion à un serveur

StrictHostKeyChecking ask

```
[root@localhost root]# ssh spirou
The authenticity of host 'spirou (192.168.225.10)' can't be established.
RSA key fingerprint is 02:a9:66:a3:73:74:e7:5c:59:fb:bc:39:5b:7a:e4:f0.
Are you sure you want to continue connecting (yes/no)? y
Please type 'yes' or 'no': yes
Warning: Permanently added 'spirou,192.168.225.10' (RSA) to the list of known hosts.
root@spirou's password: *******
Last login: Sat Mar 13 05:56:11 2004
[root@spirou root]#
```

Connexion avec message d'avertissement au début de la connexion

StrictHostKeyChecking no

```
[root@localhost root]#
[root@localhost root]#
[root@localhost root]# ssh spirou
Warning: Permanently added 'spirou,192.168.225.10' (RSA) to the list of known hosts.
root@spirou's password: *******
Last login: Sat Mar 13 05:58:49 2004 from fantasio
[root@spirou root]#
[root@spirou root]#
```

Connexion réussie sans message d'avertissement

StrictHostKeyChecking yes

```
[root@localhost root]#
[root@localhost root]# ssh spirou
No RSA host key is known for spirou and you have requested strict checking.
Host key verification failed.
[root@localhost root]#
[root@localhost root]#
```

Connexion échouant

Côté serveur : problème de clés

- Problématique lors d'une connexion ssh
  - Alerte lors d'une connexion alors que la clé enregistré dans le fichier *known_host* n'est plus la même que celle du serveur sur lequel on cherche à se connecté

```
fbongat@berlioz ~ $ ssh root@davis
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
e6:4d:f3:0e:37:09:dd:1e:d7:2b:19:18:b3:c7:88:9f.
Please contact your system administrator.
Add correct host key in /user/fbongat/.ssh/known_hosts to get rid of this message.
Offending key in /user/fbongat/.ssh/known_hosts:34
RSA host key for davis has changed and you have requested strict checking.
Host key verification failed.
```

Côté serveur : problème de clés

- Problématique lors d'une connexion ssh
  - Dans le cas d'un tel message d'alerte, il est nécessaire de *prévenir* l'administrateur du serveur afin de vérifier si ce changement a bien été volontaire
  - Ensuite afin de pouvoir se connecter, il faut supprimer l'ancienne clé dans le fichier *.ssh/known_hosts*, c'est-à-dire la ligne avec le nom de l'hôte concerné (pour tous les utilisateurs)

La clé du 192.168.225.1 a changé, supprimer cette ligne afin de ne plus avoir le message d'alerte précédent

Fichier : known_hosts

```
192.168.225.1 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAOCN/5EBOsaXMKWE4MyVKTabPD2y7RU
RMdBr2Ef1E7Ab+ajTUQMBLOzWh55znBvjcaZDJNhvy4zoQOC99lKMVOpRxpF7a5Fht+gsTj/AH4Nr279
DtadMPamBf1nUwoDKgUmZ4w9cM1c+Lid9OXwJJGbZ3Ny1t2tOiwBUd+uO7t48=
fantasio ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAwi4uT26NOpbT7cRcNvzxBwoshZ9m3GFxWAm
S3nbj34QhdNQp8Q/bjheqOeccM/vQKuNY5v9GXhp+Y4SB1vFebUXGk5gMdYiO7K6A+MMQVZzEALa1wXa
73BLKYfZGl2fARaheZ2jheMaZxuQRtZQuX899Ofe9aX9dE4oJDGtZeLM=
```

Côté serveur : clés du serveur

- Les fichiers de clés du serveur *sshd*
  - En fonction de la version de protocole et des algorithmes :
    - Les clés privées :
      - Protégés en lecture pour root uniquement
      - ssh_host_key : ssh v1, RSA1
      - ssh_host_key_rsa : v2, RSA
      - ssh_host_key_dsa : v2, DSA
    - Les clés publiques
      - En lecture pour tous
      - *.pub extension valable dans tous les cas

Côté serveur : clés du serveur

- Exemples :
  - Clé privée (rsa) d'un serveur (elle est non chiffrée)

```
-----BEGIN RSA PRIVATE KEY-----
MIICWgIBAAKBgQDEhkoS9gonVcYH8rup2kOaaIxpkyLvHnuDo4c9AVrC//C3DHhf
jqSlimXUdw33bQOtbxox2qrd3X8ZQhC3UFqEYy59EkuttyDsnUIOz36Rvbq3m2Sh
MyydAXBmuztihYC9zd83eEwxBFLpAge7x/RSzxWnrpu7ouaigviMxZn9CQIBIwKB
gDKI7nnnfvQr/7jmpUju+3DZDinGv9cWd4g//jRCLUgV7XDl8xFB79LLOQq/hz+f
sHON8Mr9tuiYCr1aIY45O9hyOHnejCg8PEQAl7Wr9Armoju7CMug/2OZ2K9wqIQJ
EOOL1gGq8ZbsjX2KytugXoCw5gVHhUOyigXzGyeV4Jf7AkEA+voguuPRkuUtNoyQ
88ETw8Yog+uIotyiSO4pR+oqzS17LEmyKXqRBD6jvqARrQpmmZDYCnbjr11bv1A5
nRTtNQJBAMh1LRJjP6vDKLm8181BU5V9ps5W5Dqni1P+r6DB4OpdiMjjftJ7cYSY
smCLqVXsMKIGSVUectmpaIQF1ajGTwUCQA5Xb5WmnkLohj2hoeILCHGWS3VAqLjZ
aFvwWiFd1o9hrORb+41XdgA+F/xDqTOWiYUeODscvI2tkDbRYmC/XgMCQEpOqldJ
b2uu4z2siqtSxOYnWzayjOj8ZvNP+BcjcMwUHNzmyLSUQBtOqKeEVNbFcSY82WjC
KqieAj+qZU1Q+McCQQCNUO+T/BiGOmtJaeyllfkIuvsIHCdxiuxUAVOyyDzopTM3
iLJgFL28QkpVGrPavcFNrRa4WaVaQsPf57B1PYL+
-----END RSA PRIVATE KEY-----
```

Côté serveur : changement de clés

- Changement de **clés du serveur**
  - Commande : ssh-keygen –t Algo –f files

    Options :

    -t : algorithme

    –f fichier cible

    -N : phrase d'identification vide

Changement de la clé rsa

```
[root@localhost root]#  ssh-keygen -t rsa -f /etc/ssh/ssh_host_rsa_key -N ""
Generating public/private rsa key pair.
/etc/ssh/ssh_host_rsa_key already exists.
Overwrite (y/n)? y
Your identification has been saved in /etc/ssh/ssh_host_rsa_key.
Your public key has been saved in /etc/ssh/ssh_host_rsa_key.pub.
The key fingerprint is:
4d:0e:9f:c0:87:71:7a:fb:cb:85:79:44:f6:3f:b8:18 root@localhost
[root@localhost root]#
```

Côté serveur : changement de clés

- Conséquence d'un changement de clés du serveur
  - Alerte lors d'une re-connexion alors que la clé du serveur a changé :
  - Alerte pour tous les utilisateurs qui chercheront à se connecter sur le serveur sshd

```
[root@localhost root]# ssh fbongat@localhost
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
4d:0e:9f:c0:87:71:7a:fb:cb:85:79:44:f6:3f:b8:18.
Please contact your system administrator.
Add correct host key in /root/.ssh/known_hosts to get rid of this message.
Offending key in /root/.ssh/known_hosts:1
Password authentication is disabled to avoid man-in-the-middle attacks.
Agent forwarding is disabled to avoid man-in-the-middle attacks.
X11 forwarding is disabled to avoid man-in-the-middle attacks.
Permission denied (publickey,password,keyboard-interactive).
[root@localhost root]#
```

Côté serveur : généralités

- sshd en écoute sur le port 22/TCP
  - Toutes les communications ssh passent par ce port (ssh, scp, sftp, tunnels ...)

```
[fbongat@vivaldi fbongat]$ netstat -na| grep ":22"
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
```

- Mode debug :
  - Mode debug du serveur sshd :

    # sshd –d
  - Sur un port non privilégié :

    (pour des tests)

    # sshd –d –p 5555

```
vivaldi ~ # /usr/sbin/sshd -d -p 5555
debug1: sshd version OpenSSH_4.6p1
debug1: read PEM private key done: type RSA
debug1: private host key: #0 type 1 RSA
debug1: read PEM private key done: type DSA
debug1: private host key: #1 type 2 DSA
debug1: rexec_argv[0]='/usr/sbin/sshd'
debug1: rexec_argv[1]='-d'
debug1: rexec_argv[2]='-p'
debug1: rexec_argv[3]='5555'
debug1: Bind to port 5555 on 0.0.0.0.
Server listening on 0.0.0.0 port 5555.
socket: Address family not supported by protocol
```

Côté serveur : démarrage/arrêt

- Lancement du serveur :
  # /etc/init.d/sshd start
  (ou # service sshd start)
- Arrêt du serveur :
  # /etc/init.d/sshd stop
  (ou # service sshd stop)
- Relancer le serveur après une modification de la configuration (sshd_config) :
  # /etc/init.d/sshd reload
  (ou # service sshd reload)
  # /etc/init.d/sshd restart
  (ou # service sshd restart)

Rechargement du fichier de configuration

Le redémarrage du serveur sshd n'interrompt généralement pas les connexions ouvertes

Authentification forte (AF)

- SSH et l'authentification forte …

Authentification forte

- Connexion par authentification forte
  - Basée sur une procédure d'identification plus complexe et différente de celle des systèmes Unix classiques *(nom et mot de passe)*
  - Cette procédure repose sur un principe d'une paire de clés publique/privée dont la clé privée est protégée par une phrase d'identification
    - **ATTENTION !** la sécurité de *ssh par authentification forte* repose alors sur la protection de la clé privée : il faut **impérativement** mettre une VRAI PHRASE D'AUTHENTIFICATION (au moins 14 caractères)
    - qui est en fait plus qu'un simple mot de passe, mais une véritable phrase (moins de limitation)
      - Permet l'utilisation de caractères blancs (séparateur) et d'autres, mais attention aux caractères utilisés à cause des différents types de claviers afin de pouvoir taper la passe-phrase
    - L'utilisateur s'identifiera alors sans utiliser le mot de passe de la connexion classique *(mot de passe Unix)*, qui lui circule sur le réseau, mais par sa phrase d'identification sur l'hôte local

Authentification forte

- Connexion par authentification forte
  - Utilise un algorithme très puissant pour le chiffrement *(algorithme RSA/DSA)*
  - Ainsi chaque utilisateur possède son propre jeu de clés uniques (une clé privée = secrète, une clé publique = accessible par tous)
  - une nouvelle connexion nécessite l'installation de la clé privée sur le client et de la clé publique sur le serveur
  - le serveur va créer un *challenge* et donner un accès au client si ce dernier parvient à déchiffrer le challenge avec sa clé privée

AF : gestion des clés - Unix

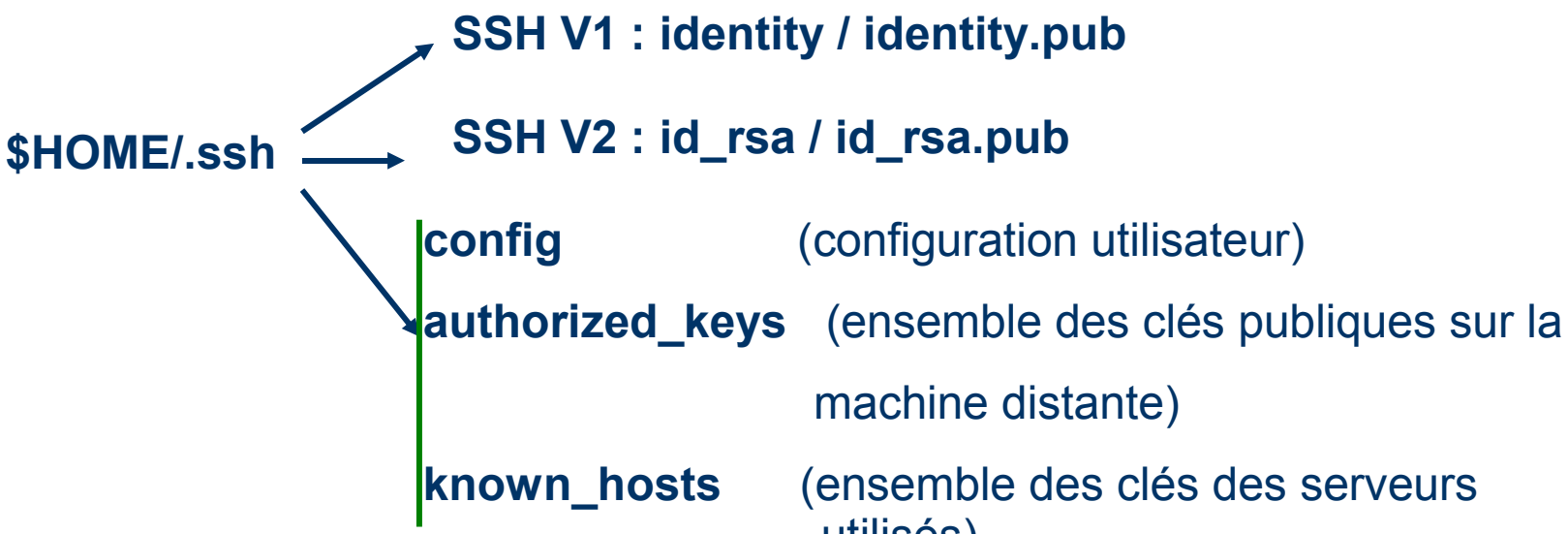
- Gestion des clés et agents = AF
  - Commandes liées :
    - Création des paires de clés :
      - ssh-keygen

      Options : **-t** algorithme : choix de l'algorithme (rsa1, rsa et dsa)

      -p changer sa phrase d'identification
    - Mise en mémoire des clés (évite la saisie répétée des phrases d'identification
      - ssh-agent
    - Chargement des clés dans l'agent
      - ssh-add

AF : gestion des clés - Unix

- Gestion des clés et agents = AF
  - Structure des fichiers impliqués :



AF : gestion des clés - Unix

- Gestion des clés :
  - Création des paires de clés (SSH V2) RSA avec un exemple :

```
[root@localhost root]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase): *******
Enter same passphrase again: *******
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
26:36:03:6d:ed:28:47:c0:6c:2f:c4:d9:5a:07:fd:bc root@localhost
[root@localhost root]#
[root@localhost root]# ll .ssh
total 8
-rw-------    1 root     root          963 mar 11 23:24 id_rsa
-rw-r--r--    1 root     root          224 mar 11 23:24 id_rsa.pub
[root@localhost root]#
```

Création d'une paire de clé (v2)

phrase d'identification

Nom des fichiers de clés

  - On peut de la même manière générer des clés avec un autre algorithme de cryptage : DSA

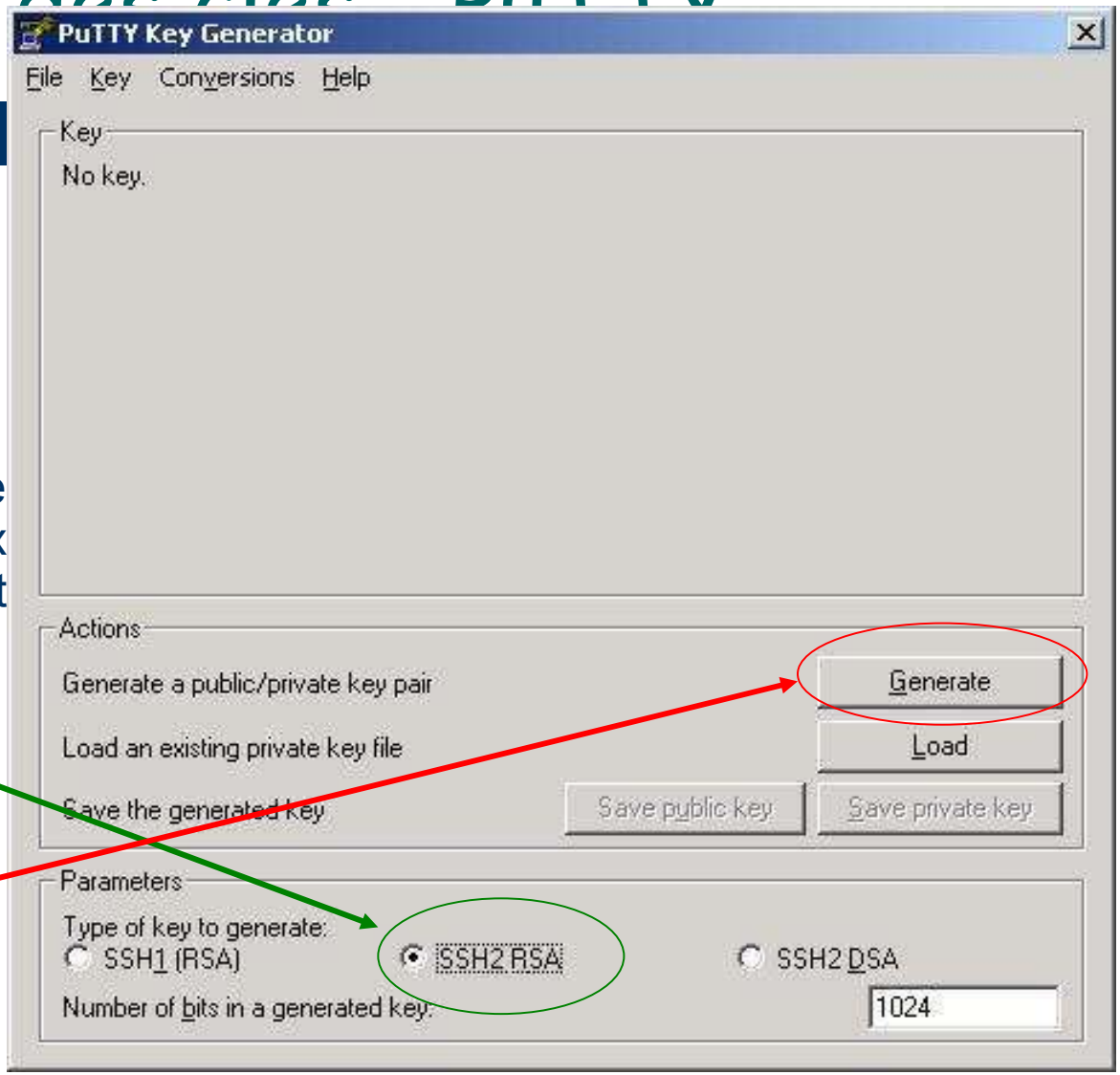
    ssh-keygen –t dsa

AF : gestion des clés - PuTTY

- Gestion des clés et agents: PuTTY
  - Création des paires de clés avec PuTTY
    - Lancer le programme *puttygen*
      - en double-cliquant sur l'icône :

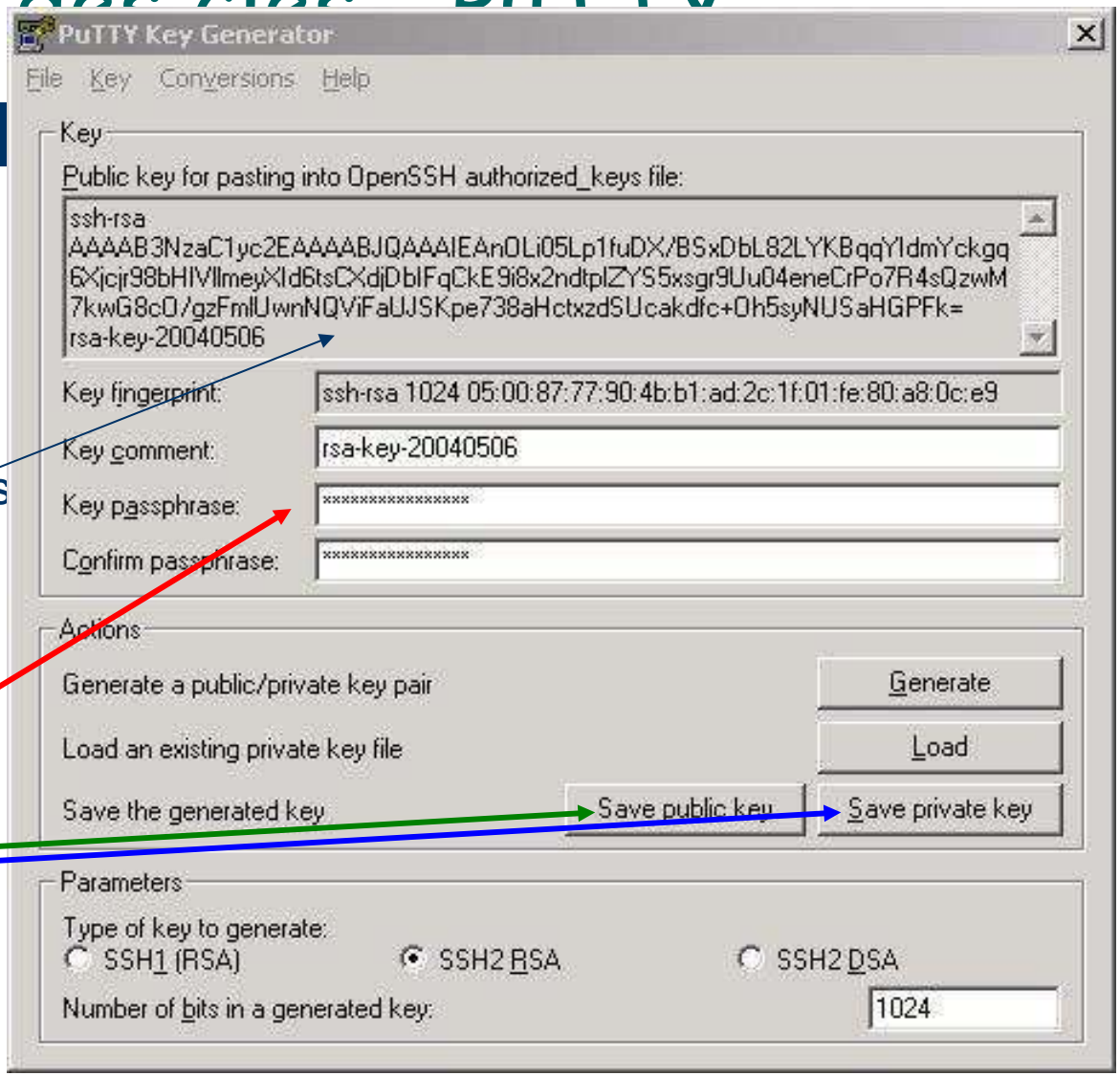
AF : gestion des clés - PuTTY

- Gestion des clés et agents: PuTTY
  - Création des paires de clés avec PuTTY
    - Sélectionner le type de clés à créer (sous Unix le plus souvent ce sont les clés v2 RSA qui sont utilisées)
    - Lancer la génération du couple de clés (publique/privée)



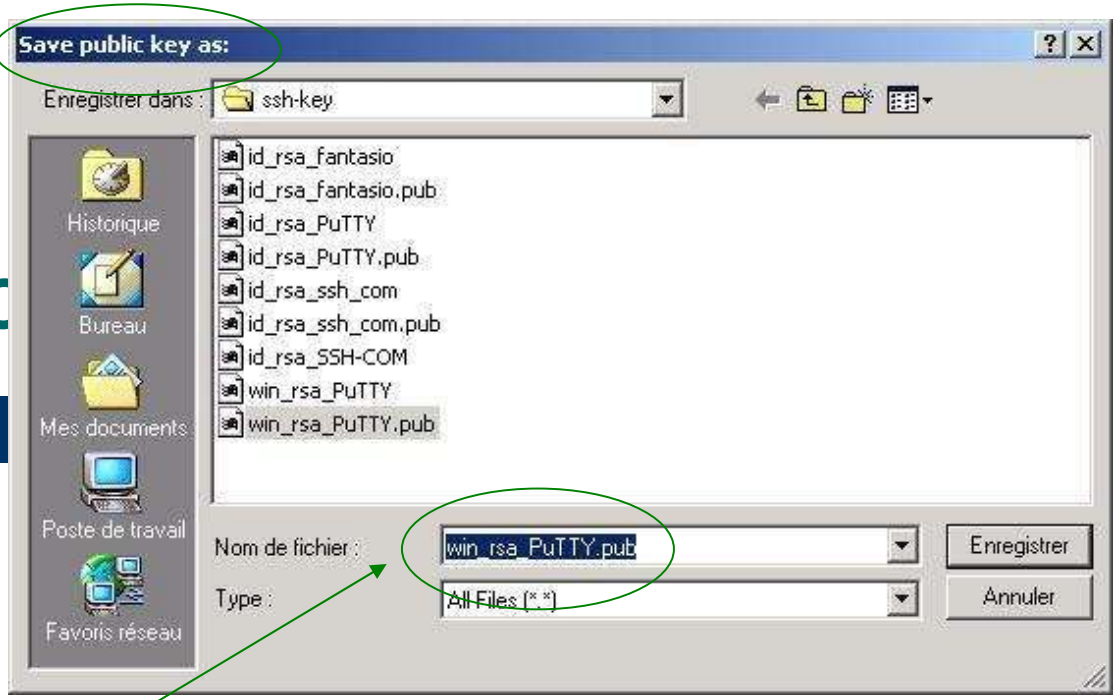
AF : gestion des clés - PuTTY

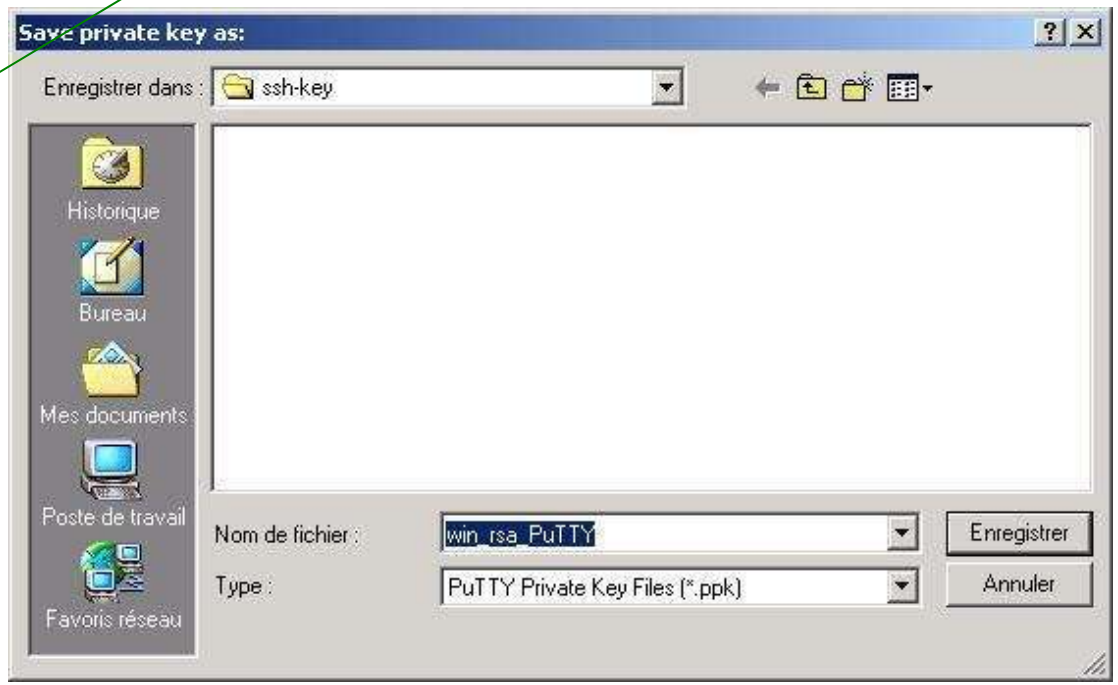
- Gestion des clés et agents: PuTTY
  - Création des paires de clés avec PuTTY
    - Les clés sont générées
    - Donner une phrase d'authentification (au moins 14 caractères minimum)
    - Sauver les clés dans 2 fichiers



AF : gestion c

- Gestion des clés et agents: PuTTY
  - Création des paires de clés avec PuTTY
    - Sauvegardes des clés
      - Donner le même nom (comme sous Unix)
      - Clé publique :
        Ajouter ***.pub*** au nom du fichier pour bien la repérer
      - Clé privée :
        Même nom sans extension .pub

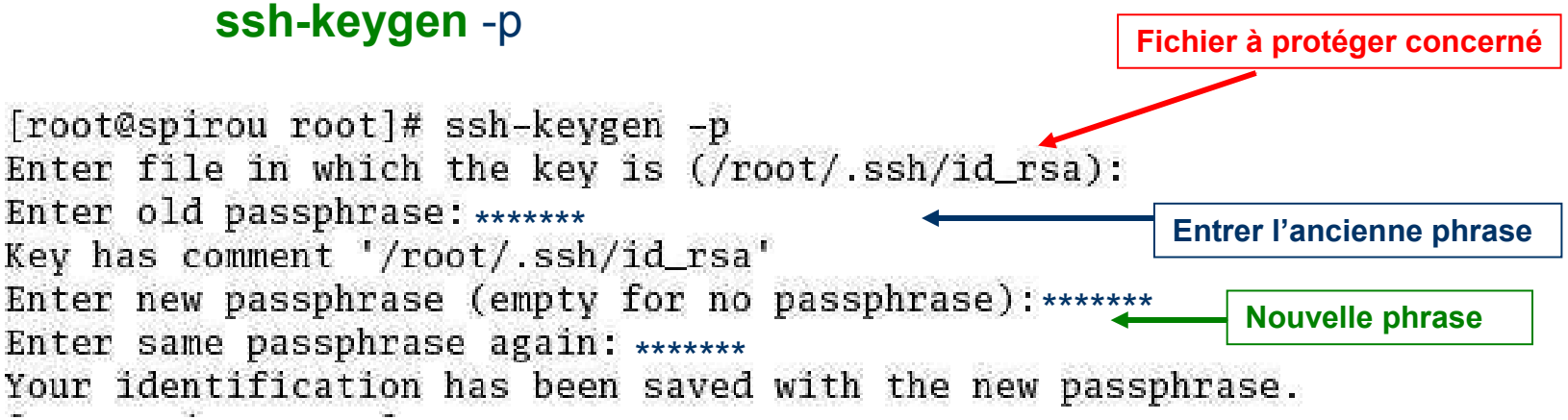




AF : gestion des clés - Unix

- Gestion des clés
  - Changement de phrase d'authentification avec un exemple :

ssh-keygen -p



AF : gestion des clés - distribution

- Distribution des clés
  - Problématique de l'installation de la clé privée (id_rsa.pub) sur le serveur distant dans le fichier des clés (authorized_keys) :
    - Soit transférer la clé publique
      - Or, Jispose de la clé publique sur la machine locale
      - Il faut la transférer et copier son contenu dans un fichier nommé authorized_keys sur la machine distante
      - Ce fichier (authorized_keys) sur la machine distante va donc pouvoir contenir plusieurs clés publiques d'utilisateurs
    - Soit envoyer le fichier (clé publique) à l'administrateur du serveur distant qui l'installera sur la machine

AF : gestion des clés - distribution

- Mise en place des clés : méthode manuelle
  - La clé privée côté client : *identity* ou *id_rsa (id_dsa)*
  - La clé publique sur le serveur dans le fichier :

    authorized_keys = clés publiques situées dans ***$HOME/.ssh***

client

serveur

$HOME/.ssh/id_rsa.pub

Transférer la clé publique sur la machine distante dans $HOME/.ssh

Puis dans $HOME/.ssh/

Faire :

cat id_rsa.pub >> authorized_keys

chmod 600 authorized_keys

AF : gestion des clés - distribution

- Mise en place des clés : méthode automatique
  - Utilisation d'un script shell : ssh-copy-id
    - installe la clé publique dans la liste des clefs autorisées (authorized_keys) d'une machine distante

client

serveur

 # ssh-copy-id *serveur*

Transférer la clé publique sur la machine distante dans $HOME/.ssh et le fichier authorized_keys directement

Rien à faire sur le serveur

AF : connexion simple Unix

- Fichier de configuration par utilisateur
  - Pour activer l'authentification forte d'un client vers un serveur lorsque celui-ci laisse tous les choix d'authentification, il suffit dans la configuration du client de configurer le fichier *config*
    - en spécifiant la variable *PubkeyAuthentication* ou *RSAAuthentication*
    - d'indiquer la clé privée utilisée (avec son path)

Fichier ~/.ssh/config

```
Host fantasio
        User fbongat
        PubkeyAuthentication yes      (ou aussi RSAAuthentication yes)
        IdentityFile ~/.ssh/id_rsa
```

AF : connexion simple Unix

- Connexions par authentification forte
  - Fichier *config*

```
[root@spirou root]# cat .ssh/config
Host spip
  User fbongat
  RSAAuthentication yes
  IdentityFile ~/.ssh/id_rsa
[root@spirou root]#
```

  - Par ssh :

```
[root@spirou root]# ssh fbongat@spip
Enter passphrase for key '/root/.ssh/id_rsa':
[fbongat@spip fbongat]$
[fbongat@spip fbongat]$
```

Phrase d'authentification

  - Par sftp :

```
[root@spirou root]# sftp fbongat@spip
Connecting to spip...
Enter passphrase for key '/root/.ssh/id_rsa':
sftp>
sftp> dir
```

AF : gestion des clés, distribution

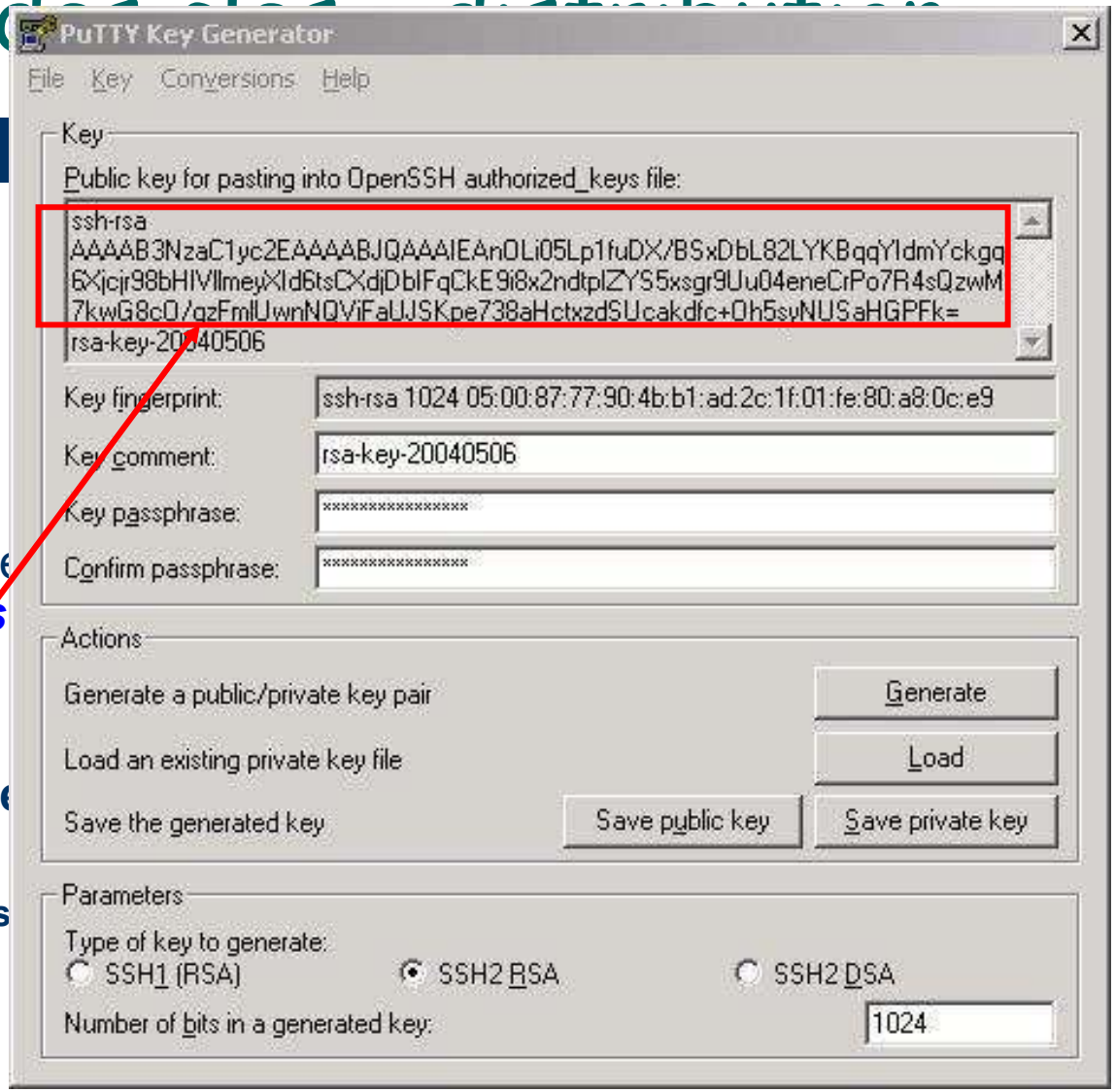
- Mise en place des clés PuTTY
  - La clé publique sur le serveur distant :
    - ***Copier/coller*** du cadre rouge sur **1 ligne** dans le fichier ***authorized_keys*** la machine distante

(ssh-rsa AAAB3…..k=)

Vérification de cohérence

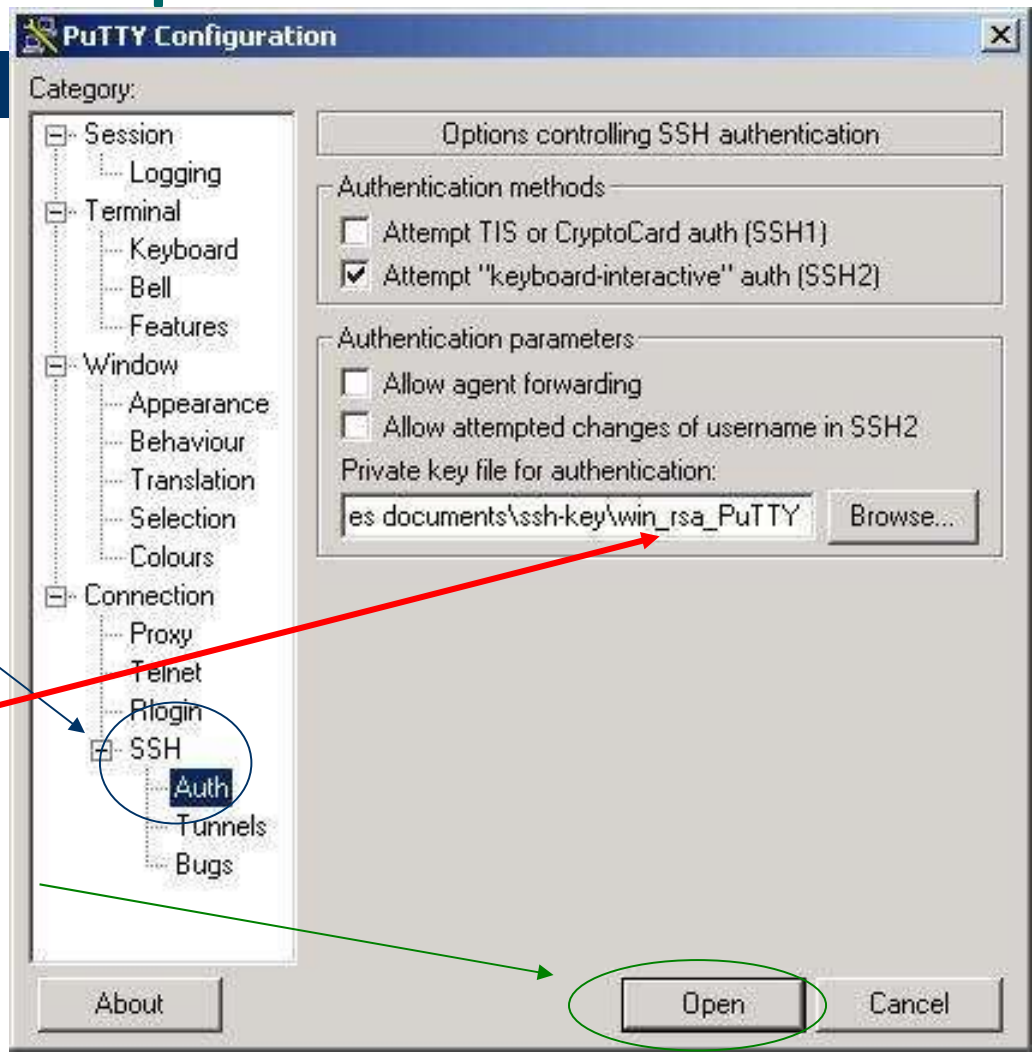
Sur linux :

ssh –l –f ~/.ssh/authorized_keys



AF : connexion simple Windows

- Connexions par authentification forte
  - Par ssh PuTTY:
    - Lancer ***putty***
    - Dans la variable ***SSH*** → ***Auth***,
    - charger la clé privée créée par PuTTY
    - Puis cliquer sur ***open***

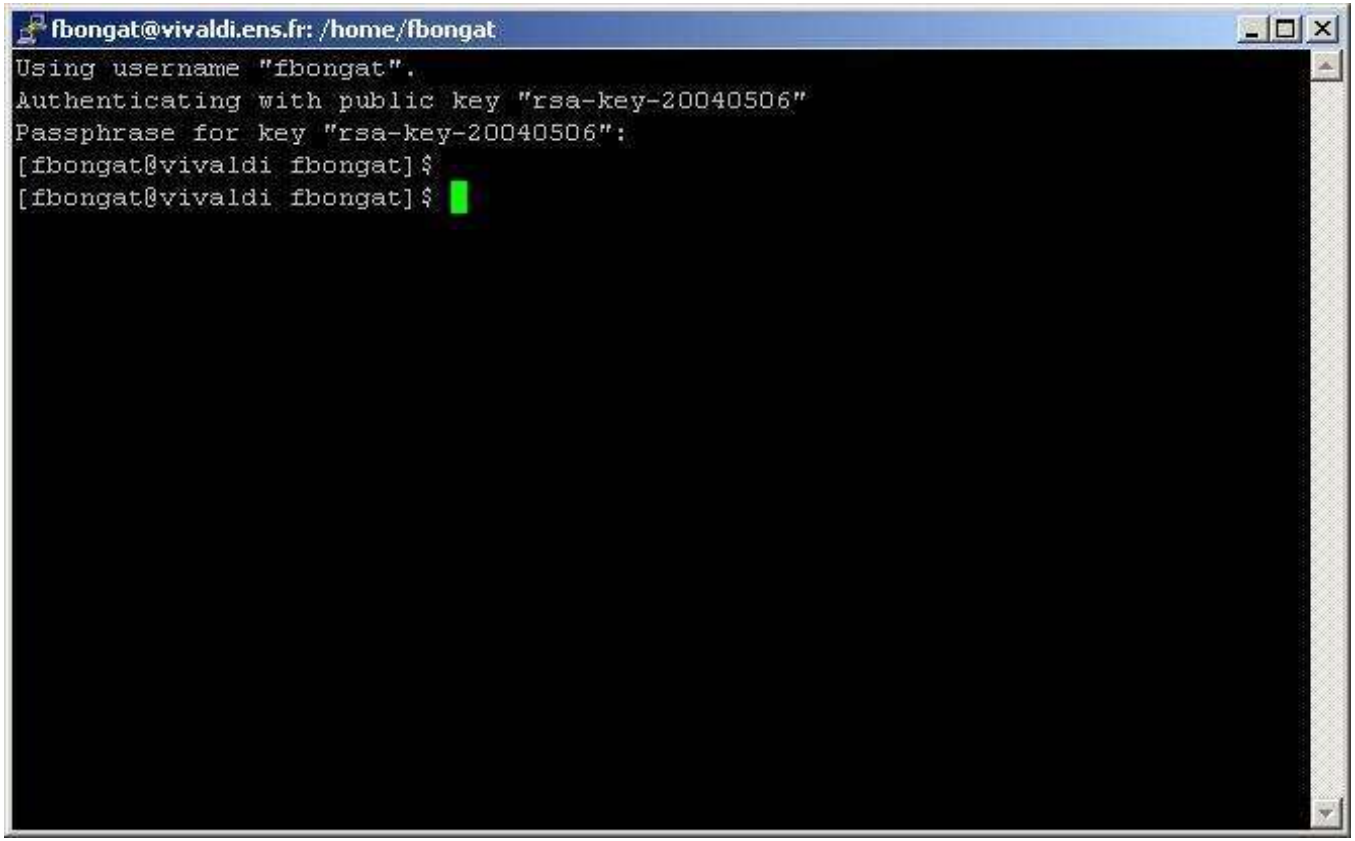


AF : connexion simple Windows

- Connexions par authentification forte

  par ssh PuTTY:

  Idem avec ***psftp*** et ***pscp***



AF : connexion via le compte root

- Gestion du super utilisateur *root*
  - Il est conseillé de ne pas permettre à l'utilisateur root de se connecter à distance

    PermitRootLogin no
  - Pour des besoins spécifiques, il est possible d'utiliser ce compte via ssh en forçant l'utilisation de l'AF

    PermitRootLogin without-password
    - La connexion n'établiera que si un clé publique de l'utilisateur est installée dans le compte root
    - Cela évite les tentatives d'attaque ssh force brute sur le compte root

AF : ssh agent

- Utilisation d'un agent pour le confort
  - Va permettre d'initier des connexions sécurisées en s'authentifiant qu'une seule fois (début d'une session ou à partir d'un terminal), et ensuite de ne plus avoir à redonner sa phrase d'authentification
  - S'effectue en 2 phases :
    - Lancement d'un agent avec la commande ***ssh-agent*** dans un shell

      # ssh-agent /bin/bash
    - puis stockage en mémoire avec la commande ***ssh-add***

      # ssh-add
    - A partir de ce moment toutes les connexions lancées de la fenêtre shell ou des xterm qui en dépendent, se font sans authentification supplémentaire.

AF : ssh agent – via un shell

- Gestion des clés et d'un agent
  - **Agent dans un shell**

```
[root@spirou root]# ssh fbongat@spip
Enter passphrase for key '/root/.ssh/id_rsa':

[root@spirou root]# ssh-agent /bin/bash
[root@spirou root]#
[root@spirou root]# ssh-add
Enter passphrase for /root/.ssh/id_rsa:
Identity added: /root/.ssh/id_rsa (/root/.ssh/id_rsa)
[root@spirou root]#
[root@spirou root]# ssh fbongat@spip
[fbongat@spip fbongat]$
[fbongat@spip fbongat]$ _
Connection to spip closed.
[root@spirou root]#
[root@spirou root]# xterm&
[1] 2704
[root@spirou root]#
```

Phrase d'identification demandée

Lancement de l'agent
Avec le shell utilisé

Ajout de la clé en mémoire

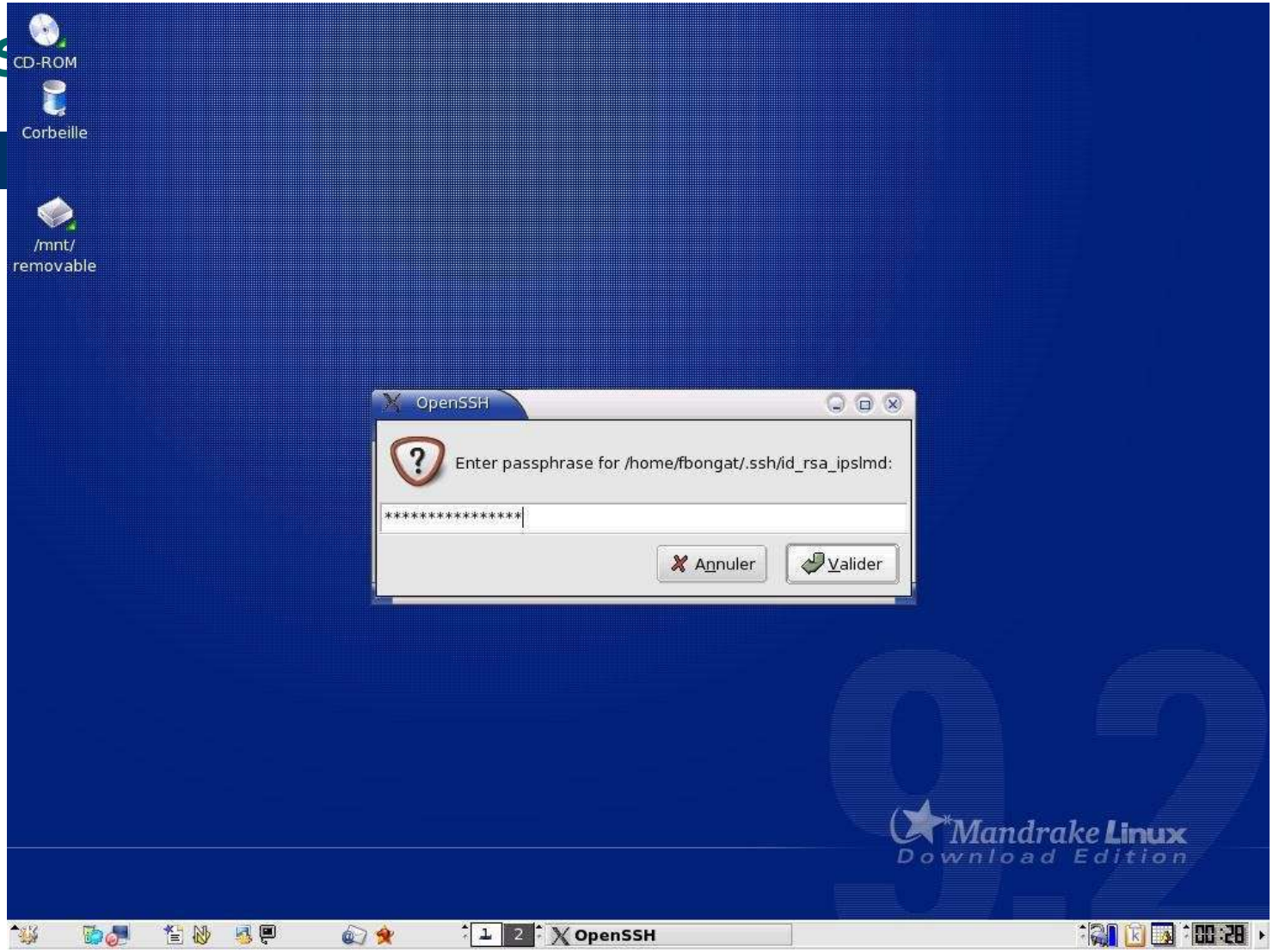
Connexion sans identification

Toutes les nouveaux terminaux lancés (xterm) à partir du terminal dans lequel on a lancé l'agent hérite de cette propriété

AF : ss

Gestion des clés et agents dans un environnement graphique

- **Lancement de l'agent et ajout de la clé privée dans l'environnement graphique (KDE)**



AF : ssh agent – session graphique

- **Keychain** : gestion des clés et des agents
  - Développé par Gentoo Linux
    - Existe pour toutes les distributions (sous forme de package)
  - Ajout de l'AF via les scripts shell de démarrage linux
  - Le programme keychain permet de réutiliser les instances de ssh-agent + ssh-add dans des sessions différentes et, si désiré, d'inviter l'utilisateur à entrer les phrases de passe à chaque ouverture de session.
    - Package à installer sur linux *keychain*
    - Lancé dans les scripts d'initialisation shell
      - */etc/profile.d/keychain.sh*

AF : ssh agent – session graphique

- Pour KDE : gestion des clés et des agents
  - Lancement dans l'environnement graphique
    - ***ssh-agent*** lancé dans /etc/X11/Xsession (par défaut rien à faire)
    - Script : **ssh-add** à rajouter dans le répertoire : ***$HOME/.kde/Autostart/***

```
[fbongat@vivaldi fbongat]$ cd .kde/Autostart/
[fbongat@vivaldi Autostart]$ ll
total 4
-rwxr-xr-x    1 fbongat  lmd-ens        83 mai 12 09:47 ssh-add*
[fbongat@vivaldi Autostart]$
[fbongat@vivaldi Autostart]$ cat ssh-add
#!/bin/bash

if [ -x /usr/bin/ssh-add ]; then
        ssh-add $HOME/.ssh/id_rsa_ipslmd
fi
```

Script ssh-add →

AF : ssh agent – session graphique

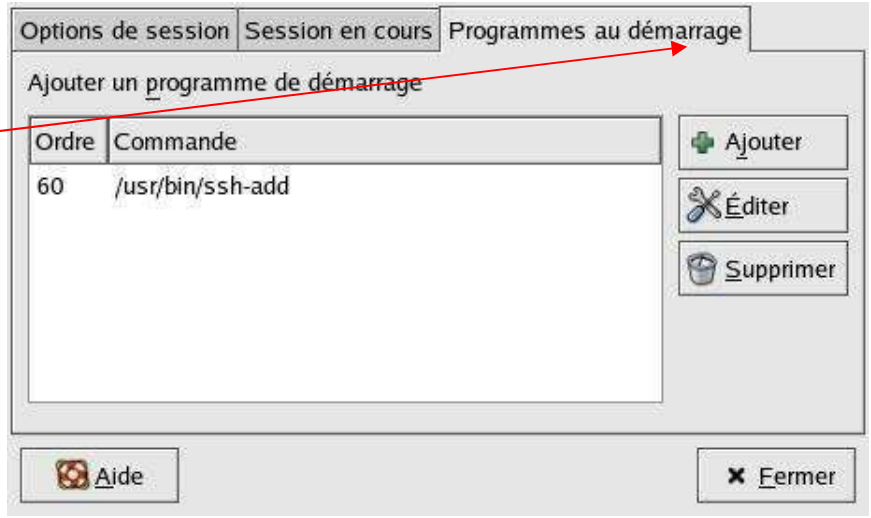
- Pour GNOME : gestion des clés et des agents
  - Autre méthode :
    - Via les menus : ***Préférences*** → ***Préférences Supplémentaires*** → ***Sessions*** → **puis l'onglet *Programme au démarrage***
    - Ou bien suivant les linux : ***Configuration*** → ***GNOME*** → ***Avancé*** → ***Sessions*** → ***et enfin l'onglet Programme au démarrage***

Ajouter un programme au démarrage d'une session cliquer sur *Ajouter*, puis entrer

/usr/bin/ssh-add

Donner une priorité de *60*

Valider



AF : ssh agent – session Windows

- Gestion des clés et agents avec PuTTY
  - Lancer l'agent en double cliquant sur l'icône ***pageant*** :

  - Il apparaît alors dans la barre des tâches actives :

  - Cliquer avec le ***bouton droit de la souris*** pour faire apparaître le menu contextuel du pageant et cliquez ensuite sur « ***Add Key*** » pour ajouter les clés privées gérées par l'agent ssh

AF : ssh agent – session Windows

- Gestion des clés et agents avec PuTTY
  - Cliquer ***AddKey***
  - Charger la clé privée Putty
  - Entrer la phrase d'authentification

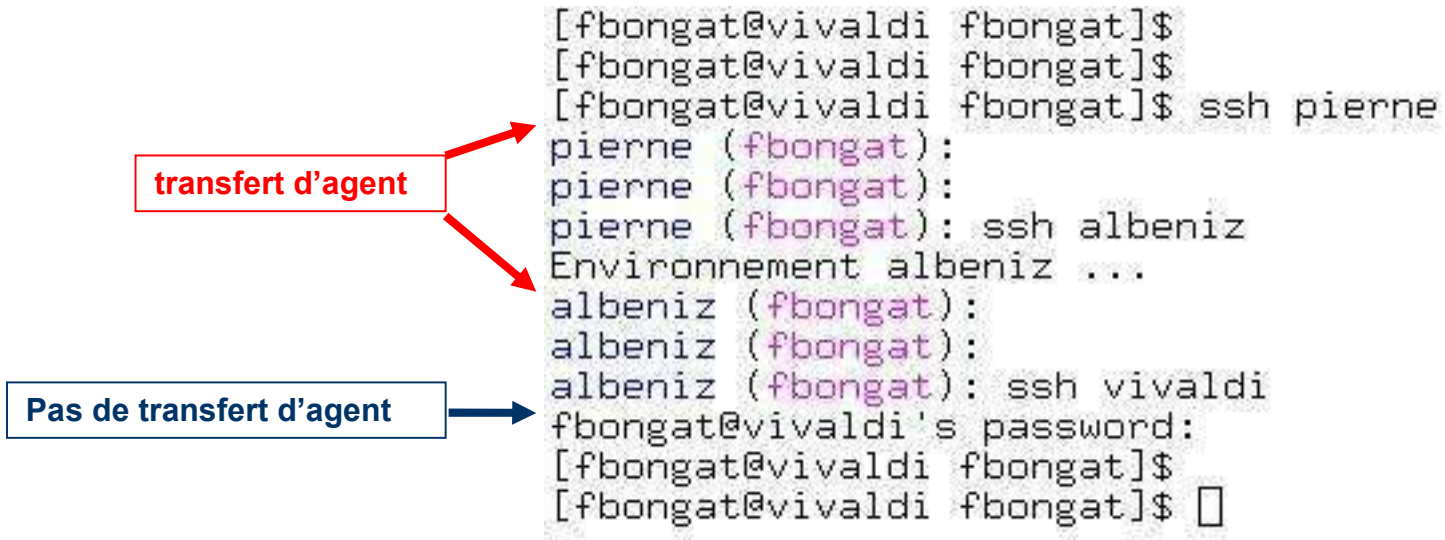
AF : transfert d'agent - Unix

- Transfert d'agent
  - relais des demandes d'authentification entre hôtes
    - Permet de cascader les hôtes sans avoir à donner d'authentification supplémentaire
    - Nécessite une configuration administrateur sur toutes les machines à cascader.
      - Il faut démarrer un agent au départ de la machine cliente source, et activer le chargement des clés en mémoire
      - Et activer le transfert d'agent au niveau de tous les serveurs concernés dans le fichier de configuration « client »

        /etc/ssh/ssh_config : ForwardAgent yes

AF : transfert d'agent - Unix

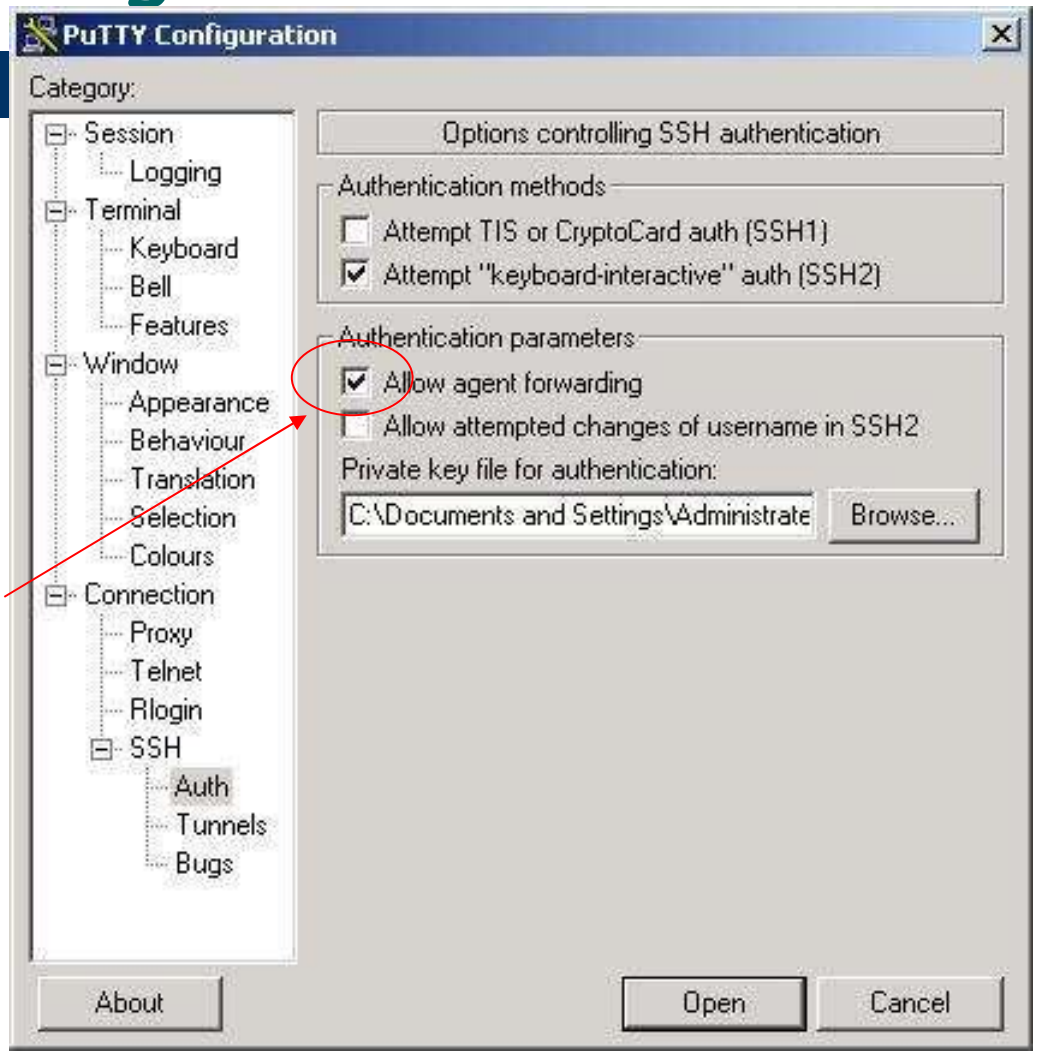
- Transfert d'agent
  - relais des demandes d'authentification entre hôtes



AF : transfert d'agent - Windows

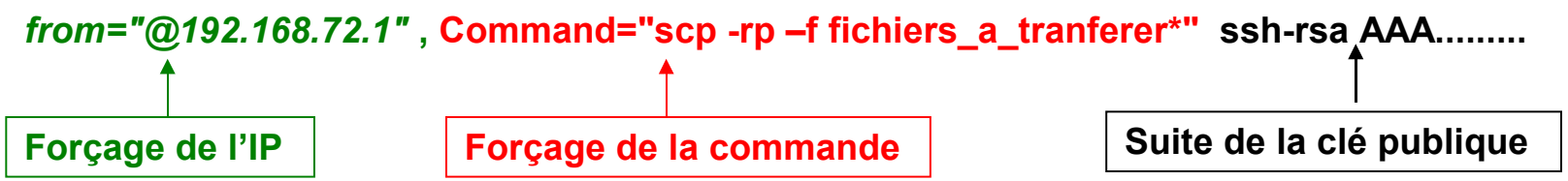
- Transfert d'agent
  - relais des demandes d'authentification entre hôte à partir d'un client PuTTY
    - Cocher la case :

      Allow agent forwarding



AF : batch ou AF sans mot de passe

- Forçage d'une commande (ou script)
  - Utilisation pour le batch : lancement de scripts de manière autonome et sans mot de passe
    - Nécessite une clé publique sans passphrase associée
      - ATTENTION cela va à l'encontre de la sécurité !!!
    - Il faut donc renforcer la sécurité par :
      - **Forcer l'exécution de la commande**
        - *exemple scp –rp –f fichier depuis 192.168.72.1 (machine source)*
      - **Forcer l'adresse IP émettrice pour forcer la source**
        - Sur la machine cible, dans le fichier authorized_keys



AF : batch ou AF sans mot de passe

- Mode « Hostbased »
  - C'est une authentification par hôte de confiance
    - On utilise une paire de clés publique et privée pour établir une connexion sécurisée et de confiance
      - Il faut cependant faire attention à ne pas se faire voler sa clé privée ou sa machine
    - Donne l'accès sans passphrase à tous les utilisateurs d'une machine de confiance vers une autre.
    - Utilisation pour le batch notamment pour les fermes de calcul
      - *Attention aux accès à la frontale --> problème de sécurité ensuite*

AF : batch ou AF sans mot de passe

- Mode « Hostbased »
  - Configuration nécessaire sous Linux :

Récupération et installation des clés des machines de confiance

```
# ssh-keyscan -t rsa machine1 machine 2 ...  > /etc/ssh/ssh_known_hosts
# cat /etc/ssh/ssh_known_hosts | cut -d" " -f1 > /etc/hosts.equiv
```

Fichier ssh_config

```
        HostbasedAuthentication yes
        EnableSSHKeysign yes
```

Fichier sshd_config

```
        RhostsRSAAuthentication yes
        HostbasedAuthentication yes
# vérifier les droits : chmod u+s /usr/lib/ssh/ssh-keysign
```

Attention aux problèmes de sécurité !!

Tunneling

- C’est une méthode d'utilisation de SSH pour sécuriser les autres applications TCP sensibles.
  - Son principe est de créer un tunnel chiffré entre deux machines et d'y faire passer les applications dedans. Ainsi, on peut continuer à utiliser des outils comme FTP, IMAP, POP, smtp ... de manière sécurisé (pour FTP: cela dépend du paramétrage du serveur ftp)
- Au niveau sécurité, les tunnels posent quelques soucis car les ports locaux sur lesquels des transferts sont établis sont accessibles à tous les utilisateurs de la machine (notamment les passerelles)

Tunneling

- Transfert de port local
  - Position du problème :
    - Il s'agit donc de rediriger des services souhaités et que l'on ne peut accéder normalement car ils sont filtrés; et d'utiliser la connexion ssh (seule acceptée) pour y faire passer ces services

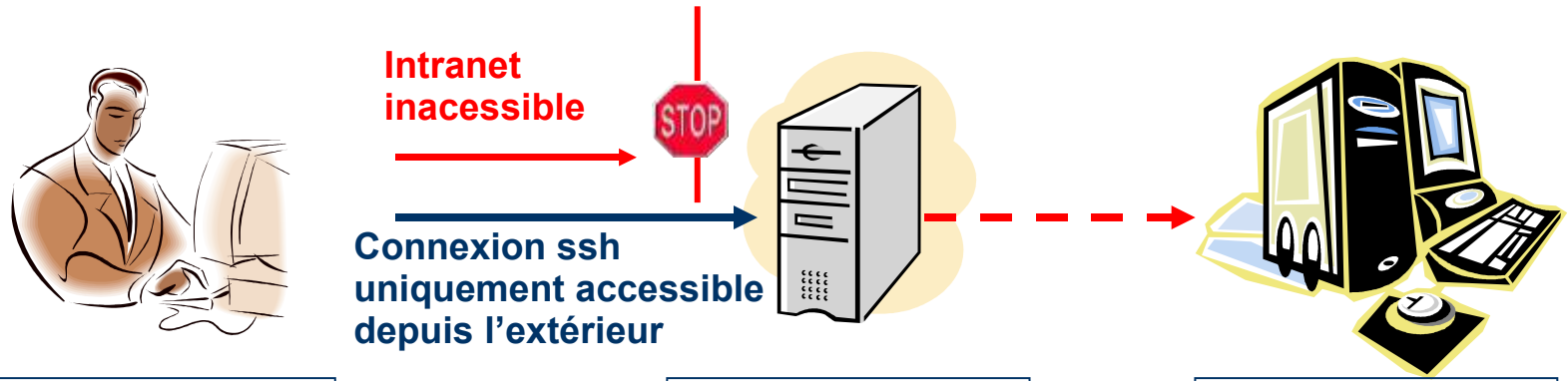
Tunneling : cas d'un intranet

- Consultation d'un Intranet depuis l'extérieur

  Créer le tunnel sur la machine locale:

  ssh –L 18000:login@ServWeb:80 passerelle

  Puis lancer le navigateur Web avec l'URL : http://localhost:18000

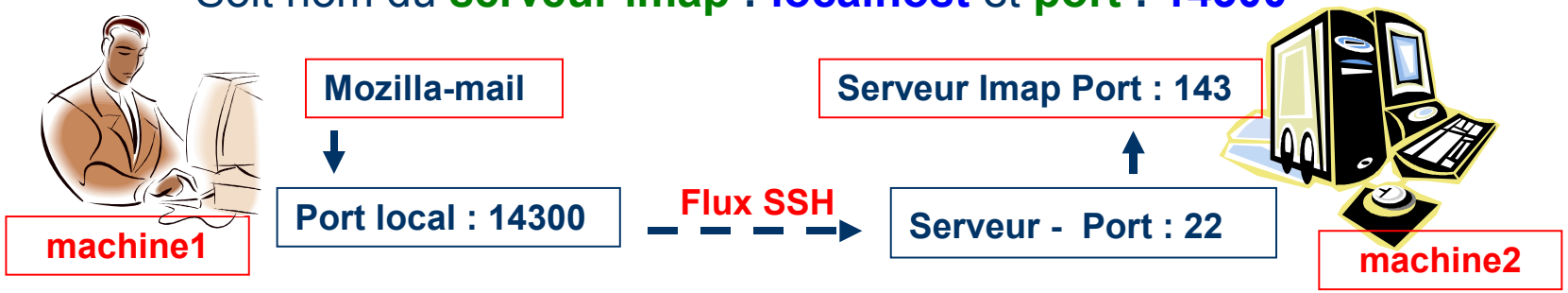


Tunneling : services communs

- Consultation de sa messagerie par Imap
  - Création du tunnel pour accéder au serveur Imap par le tunnel ssh

  ssh –L 14300:localhost:143 machine2

  ou : **ssh –N –L 14300:localhost:143 machine2**
- Puis configurer l'application client imap sur localhost:14300
  - Soit nom du **serveur imap : localhost** et **port : 14300**



-L : spécifie que les données arrivant sur le port 14300 de la machine1 doivent être envoyées sur le port 143 l'hôte distant (machine2)

-N : permet de pas donner de prompt après la connexion (ou mettre le sleep en secondes)

Tunneling : en

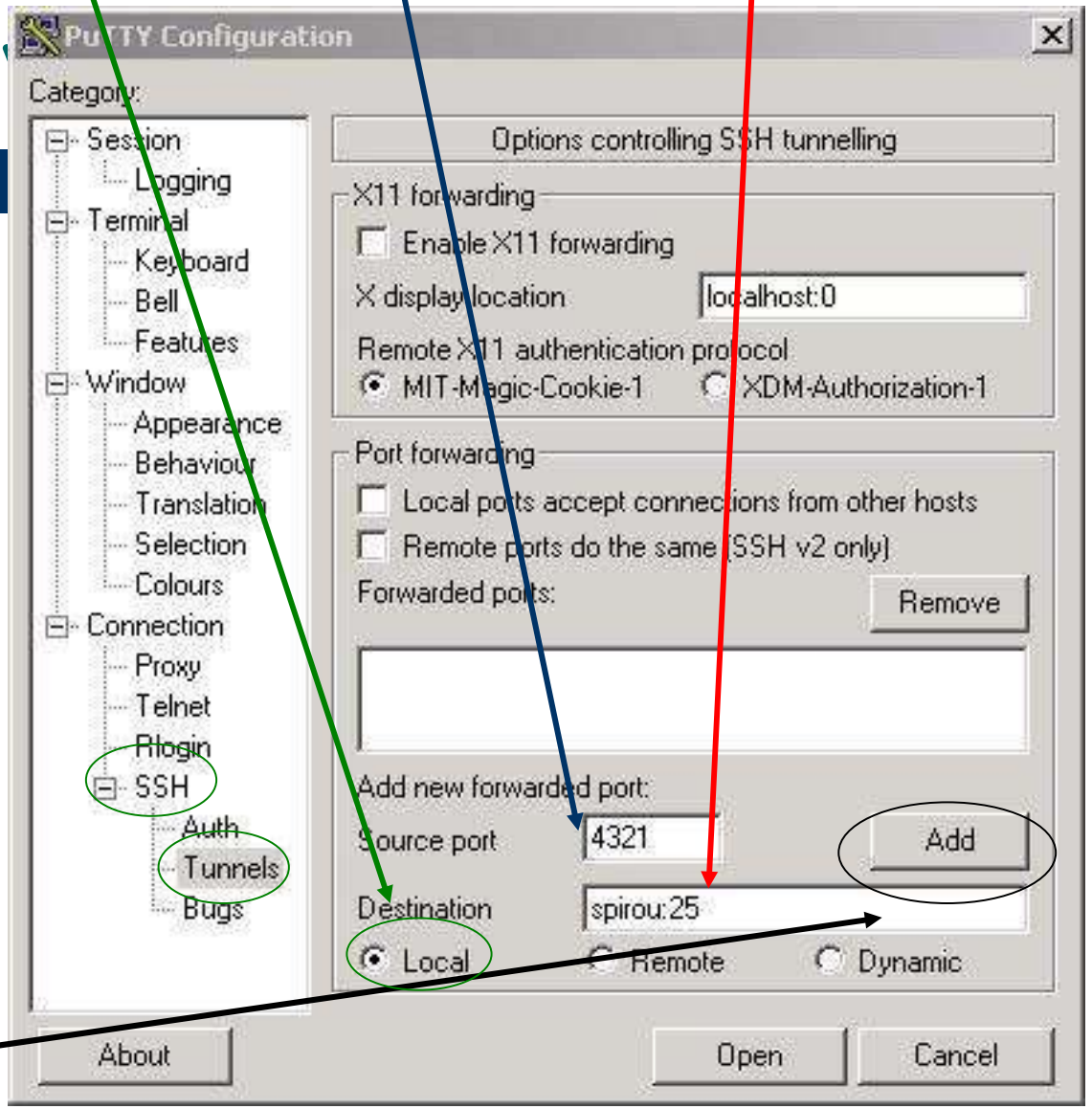
- PuTTY et smtp
  - Envoi d'un message vers un relai smtp filtré via une passerelle ssh
    - Lancer putty
    - Aller dans la valeur ***SSH*** → ***Tunnels***
    - Remplir les champs liés au tunneling dans la fenêtre
    - Puis valider cliquant sur ***Add***

Le port local (> 1024), ici : 4321

Connexion sur localhost

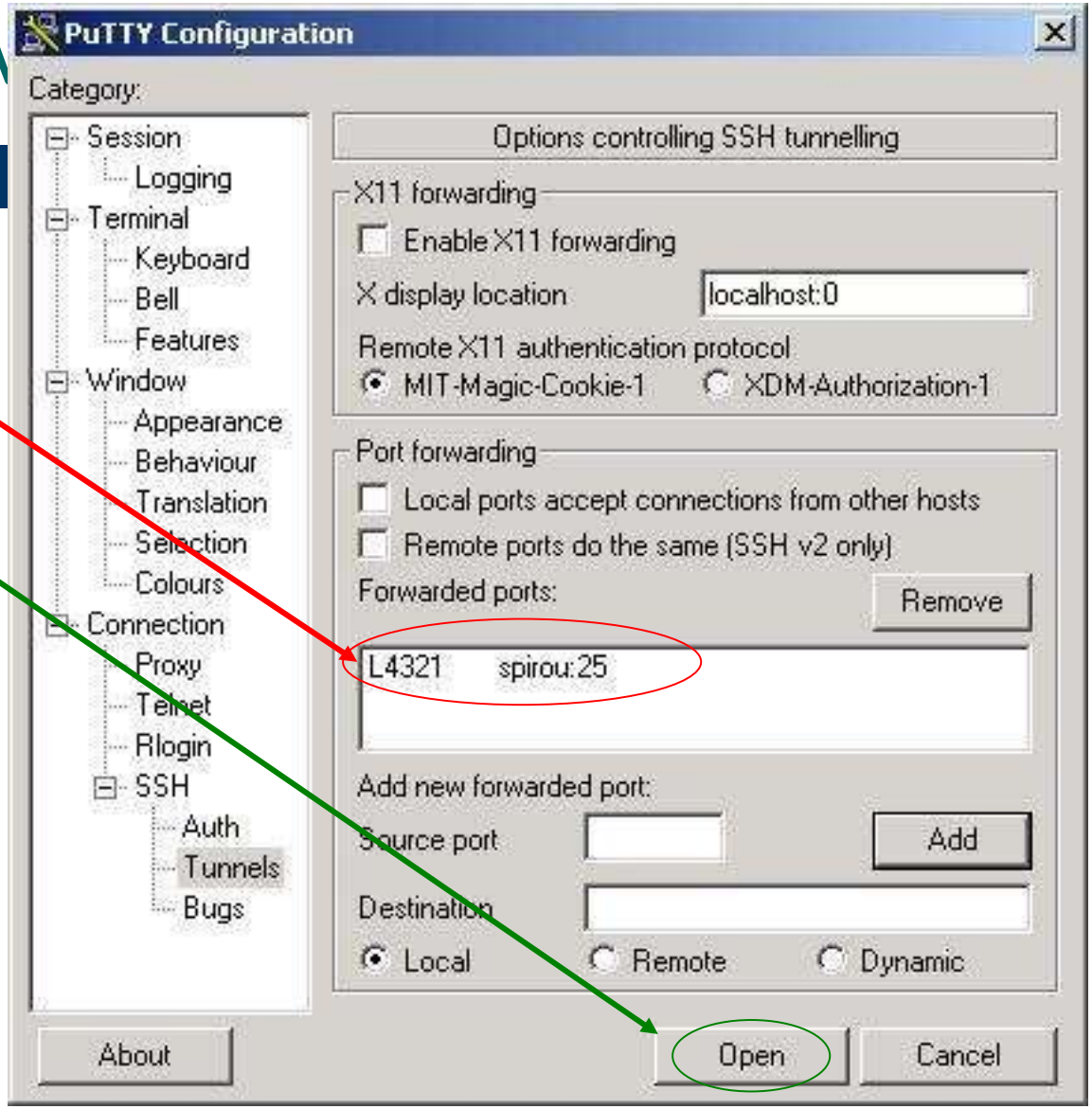
Nom de la machine distante et le port distant

Syntaxe : spirou:25



Tunneling : env

- PuTTY et smtp
  - Lancement du tunnelssh :
    - Le tunnel est configuré
    - Il faut maintenant lancer la connexion « ***Open*** »
    - Donner son mot de passe afin d'obtenir une connexion ssh classique
    - Le tunnel sera ainsi prêt
    - Configurer l'application pour qu'elle se connecte sur :
      - ***Nom distant*** : ***localhost***
      - ***Port*** : ***4321***



Tunneling : envoi de messages smtp

- PuTTY et smtp
  - Exemple smtp : simulation d'un client de messagerie
    - Configurer l'application pour qu'elle se connecte sur :
      - ***serveur smtp sortant*** **:** ***localhost***
      - ***port*** **:** ***4321***

Exemple en simulant une connexion d'un client de mail par un *telnet* en localhost sur le port 4321

```
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrateur>telnet localhost 4321_
```

Le serveur répond enfin ! On peux envoyer des messages depuis ce serveur bien qu'il soit filtré

```
220 mailhost.bdnet ESMTP Postfix
_
```

Conclusion

- SSH est une boîte à outils complète
- Les connexions sont sécurisées
  - Améliore la sécurité mais il permet aussi de la contourner
- Existe différentes possibilités de connexions et de transferts de fichiers
- Les relais possibles des applications TCP permettent d'accéder à de nouvelles ressources
- Installé en standard sur tous les systèmes (client et serveur) Unix et MacOS X
- De très bons clients Windows notamment PuTTY, Winscp notamment en les associant à Xming