SSH : Secure SHell

De l'utilisateur à l'administrateur

Version du 21/10/2004

Frédéric Bongat
[fbongat@lmd.ens.fr]

Index

Index

Index

Présentation

- Connexions à distance : Secure Shell
  - SSH est un protocole, devant sécuriser les communications
    - En fait, SSH chiffre et compresse un tunnel de session 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.
    - 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)

Présentation

- Terminologie de SSH :
  - C'est un protocole (2 versions) :
    - Version 1 et version 2
  - C'est aussi un produit :
    - SSH Communications Security Inc (V1 et V2)
      Initialement développé par Tatu Ylönen
    - OpenSSH du projet OpenBSD (V1 et V2)
      Produit en accord avec la législation française sur la cryptographie
      http://www.ssi.gouv.fr/fr/reglementation/index.html#produits_crypto
    - Et anciennement SSF (~ SSH V1) :
      d'usage libre en France et adapté à la législation française à l'IN2P3 par Bernard Perrot
  - C'est aussi une commande

Présentation

- Fonctionnement sur le schéma d'un système client – serveur

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

- 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
  - Les fichiers de configuration (OpenSSH) sont souvent dans:
    - Pour le serveur : /etc/ssh
    - Pour les clients : /etc/ssh et $HOME/.ssh

Présentation

- Clients/serveurs multi plates-formes :
  - Windows (clients gratuits) :
    - ssh.com (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, la version Tectia 4 est payante. 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
      - Outil graphique de transfert de fichiers (scp/sftp, donc pas de ssh) très performant
      - http://winscp.sourceforge.net/eng/
    - 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
      - 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

Présentation

- L'authentification
  - 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

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 : voir page 19)

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]#
```

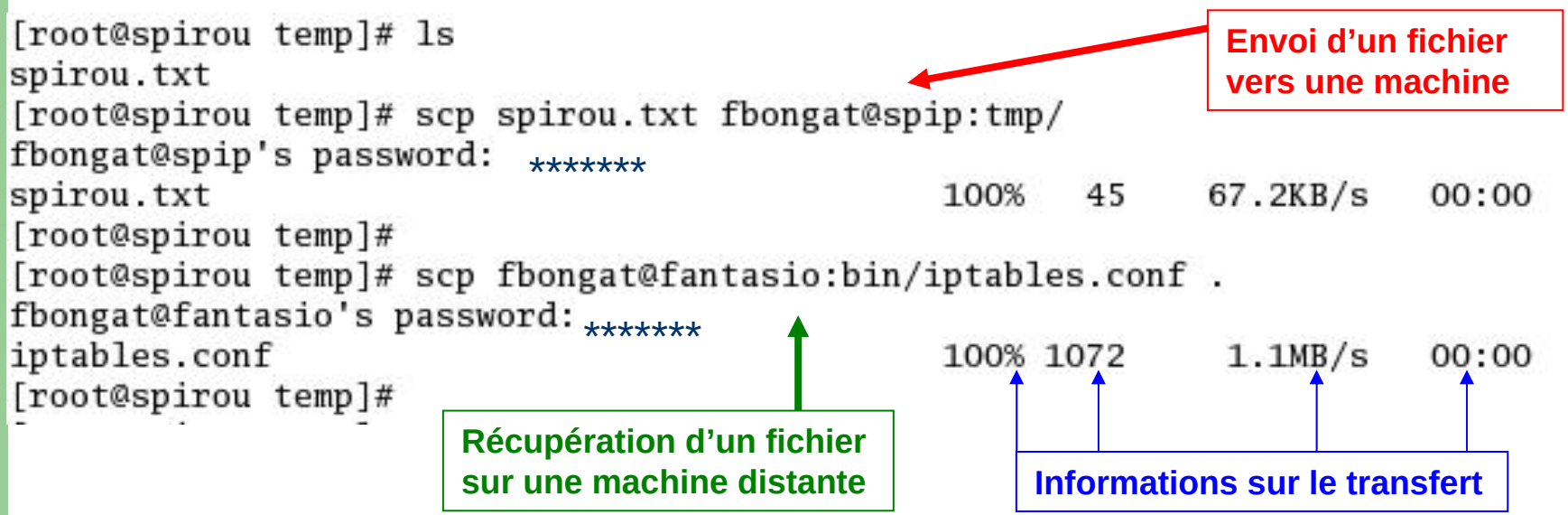
Côté client : unix - sftp

- 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>
```

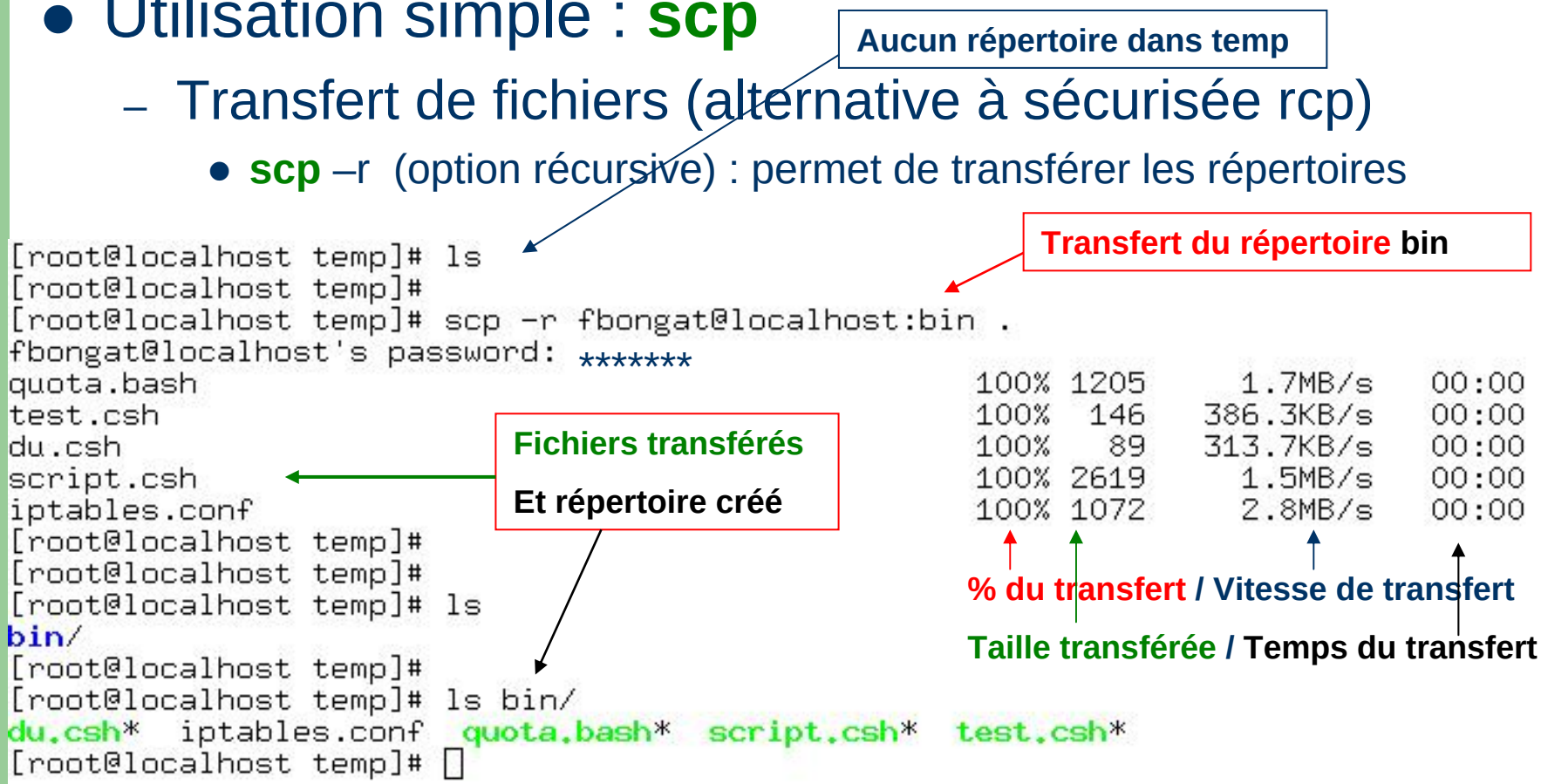
Côté client : unix - scp

- 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



Côté client : unix - scp

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



Côté client : unix - options

- Principales options :
  - ssh en mode débug (de plus en plus verbeux) :
    - Niveau 1 : **ssh –v**
    - Niveau 2 : **ssh** –vv
    - Niveau 3 : **ssh** -vvv
  - Connaître la version de ssh : **ssh -V**

```
[fbongat@vivaldi fbongat]$ ssh -V
OpenSSH_3.6.1p2, SSH protocols 1.5/2.0, OpenSSL 0x0090702f
```

Version 1 et 2

  - Activer la compression :

    ssh –C login@machine

  - Activation d'un port privilégié pour les connexions sortantes (lors d'une connexion passant par un firewall qui n'autorise pas l'utilisation des ports privilégiés <1024)

    ssh –P login@machine

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 fantasio

        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 fantasio (à la place de ssh fbongat@fantasio)

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

Côté client : sans mot de passe

- Connexions sans mot de passe :
  - Relation de confiance entre deux machines et deux utilisateurs
    - Basé sur un fichier **.shosts** (équivalence ssh de rsh avec son .rhost)
    - Permet de faire des scripts en background
  - Doit être autorisé (et configuré) par *l'administrateur* du serveur ssh
  - Doit être paramétré par *l'utilisateur* du client ssh (.shosts et clé publique)
  -  **!** Risque pour la sécurité (les comptes utilisateurs concernés sont plus sensibles)

Côté client : sans mot de passe

- Connexions sans mot de passe :
  - Mise en œuvre :
    - exemple : spip (hôte source) → spirou (hôte distant)
      - Configurer sur la machine distante le fichier *.shosts* :

```
[fbongat@spirou fbongat]$ vi .shosts
[fbongat@spirou fbongat]$ chmod 600 .shosts
[fbongat@spirou fbongat]$ ll .shosts
-rw-------    1 fbongat  user          19 avr 13 18:35 .shosts
[fbongat@spirou fbongat]$
[fbongat@spirou fbongat]$ cat .shosts
spip.bdnet fbongat
[fbongat@spirou fbongat]$ _
```

Attention au droit du fichier : seulement l'utilisateur qui en est le propriétaire

Contenu du fichier .shosts : nom fqdn obligatoire (hostname+domain)

Côté client : sans mot de passe

- Connexions sans mot de passe :
  - Mise en œuvre :
    - Installer la clé publique de la machine locale sur la machine distante le fichier *.ssh/known_hosts* :
      - La clé doit être obligatoirement associée au nom de la machine sans domaine et avec domaine (si il n'y a que la clé avec le hostname) et son adresse ip

C'est la clé avec le nom + fqdn (hostname+domain) + IP qui est utilisée

```
[fbongat@spirou .ssh]$ cat known_hosts
spip,spip.bdnet,172.16.158.20 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAw2TR9mXxNludjK
TIQN4RX2NkAUBlOM716GeEmkTUgP3dqj4+2Klg+p15soJseACoZIqnDAZizmiLvnePtd3Gp3wDp4GqZ+
YsrAjMMzTKDEWJh+KzF+4ZuNXAlBgjSvgLOjRYWLJz1EvOxPZ6KCfovpnrDp+O2TUIeAIjxl/23iM=
```

Côté client : sans mot de passe

Pas de mot de passe demandé

- **Connexions sans mot de passe** :
  - **Mise en œuvre** : *exemple*

    spip → spirou

```
[fbongat@spip fbongat]$
[fbongat@spip fbongat]$
[fbongat@spip fbongat]$ ssh spirou
[fbongat@spirou fbongat]$
[fbongat@spirou fbongat]$
```

  - **Mode debug (ssh –v) :**

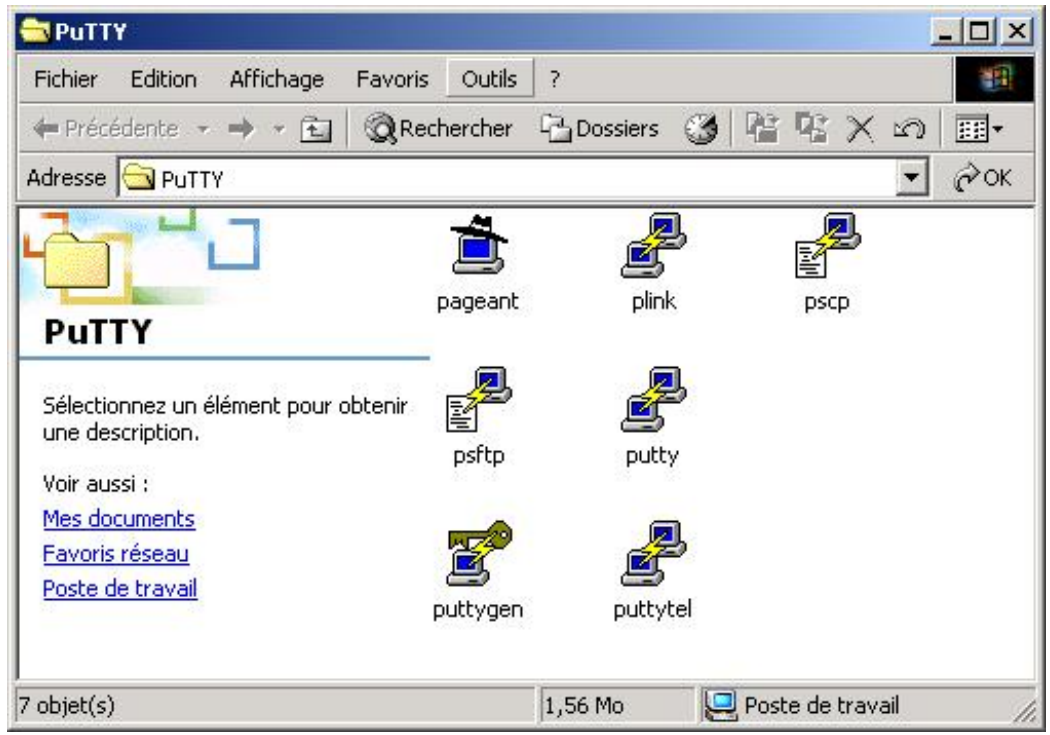
```
debug1: Found key in /home/fbongat/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password,keyboard-interactive,hostbased
debug1: Next authentication method: hostbased
debug1: Remote: Accepted by .shosts.
debug1: Remote: Accepted host spip.bdnet ip 172.16.214.20 client_user fbongat server_user fbongat
debug1: Authentications that can continue: publickey,password,keyboard-interactive,hostbased
debug1: Remote: Accepted by .shosts.
debug1: Remote: Accepted host spip.bdnet ip 172.16.214.20 client_user fbongat server_user fbongat
debug1: Authentication succeeded (hostbased).
```

Vérification du bon fonctionnement : .shosts est accepté

Côté client : PuTTY - ssh

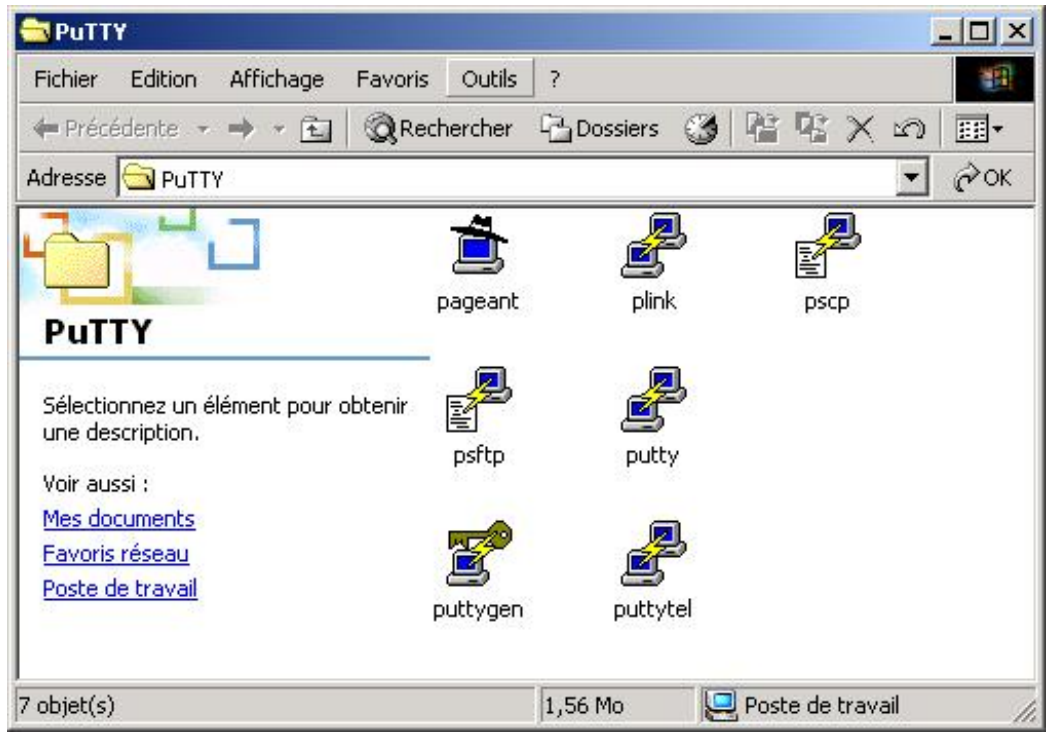
- SSH et Windows : PuTTY
  - Implémentation libre
  - Proche d'OpenSSH
    - Boîte à outils qui comprend un ssh, sftp, scp, ssh-agent et utilise des clés (compatibles avec les clés OpenSSH)
    - 7 binaires (ou un fichier zip) dont 5 indispensables (pageant, pscp, psftp,putty et enfin puttygen) à copier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

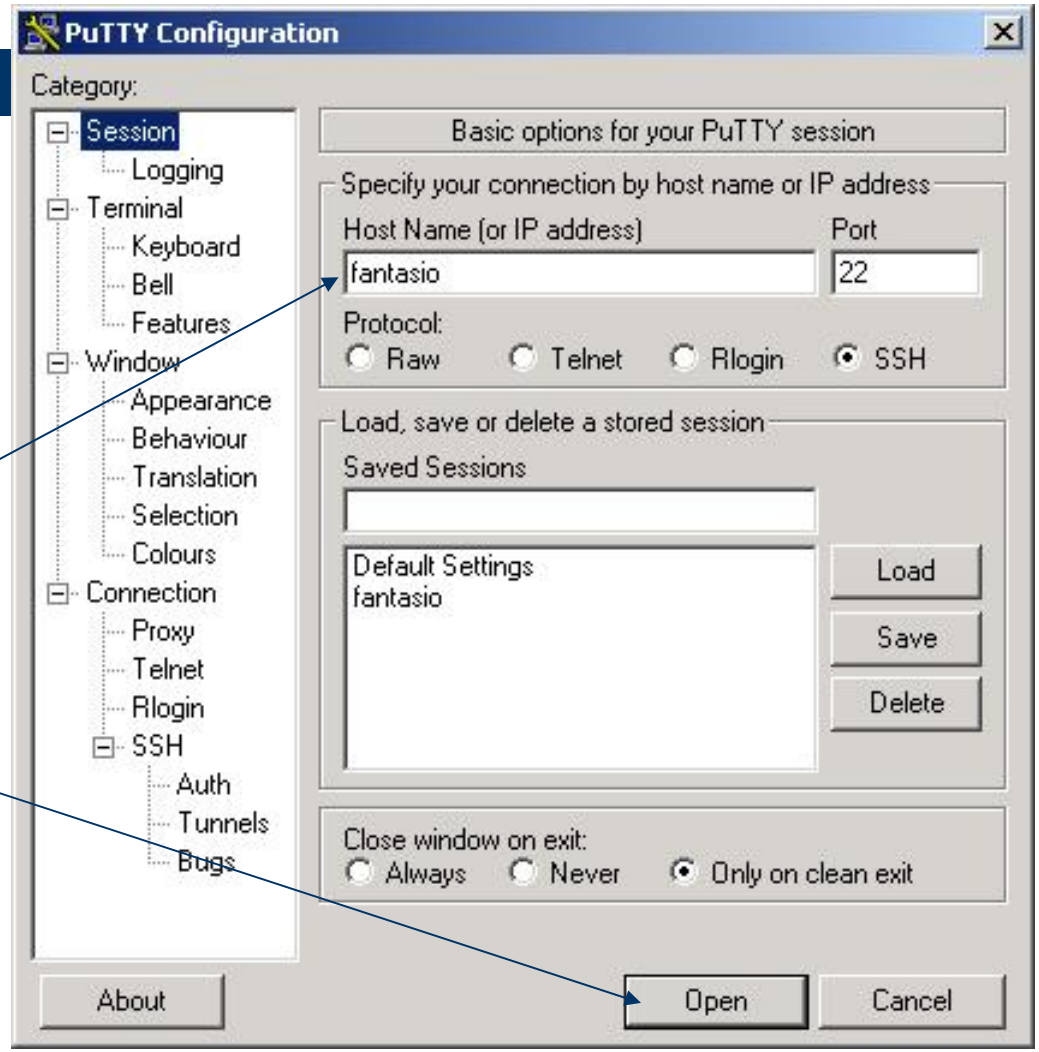


Côté client : PuTTY - ssh

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

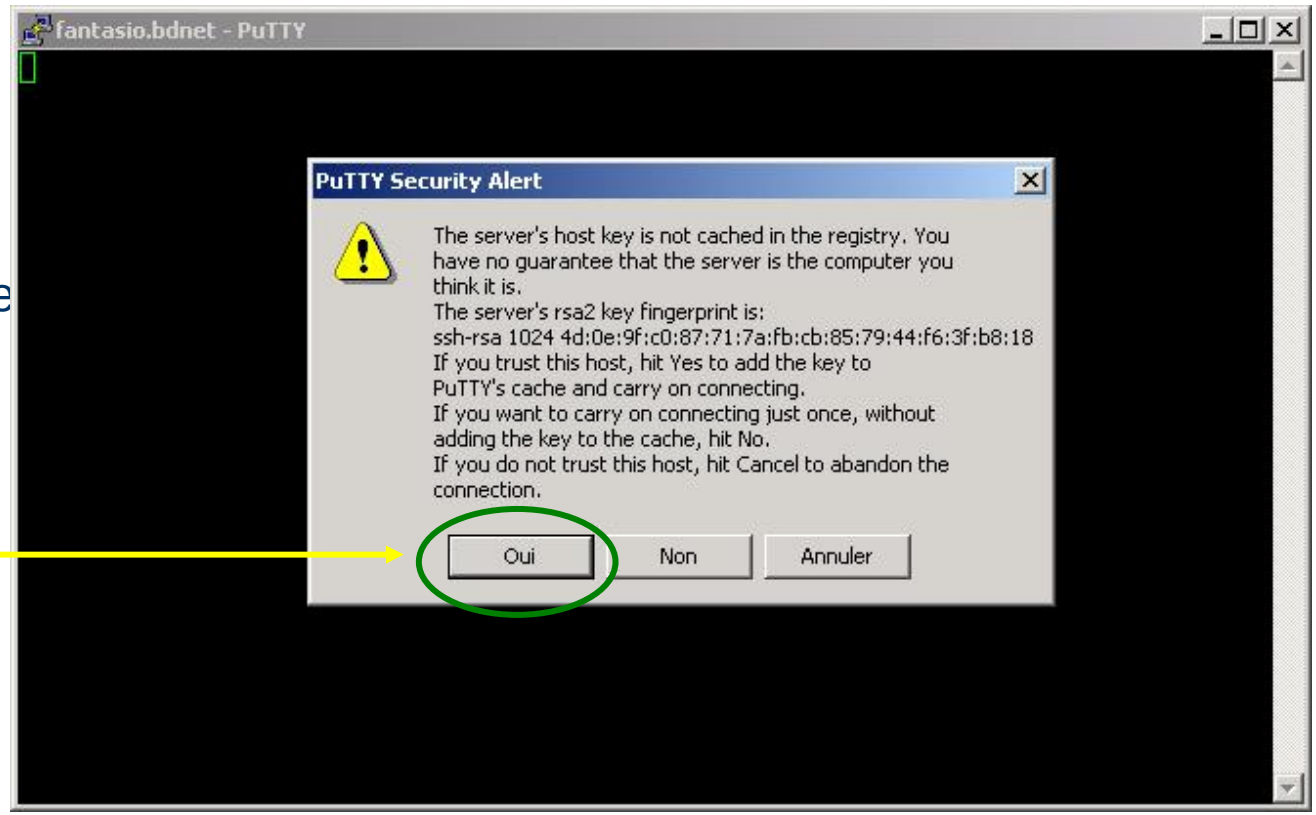
        Host Name

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



Côté client : PuTTY - ssh

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

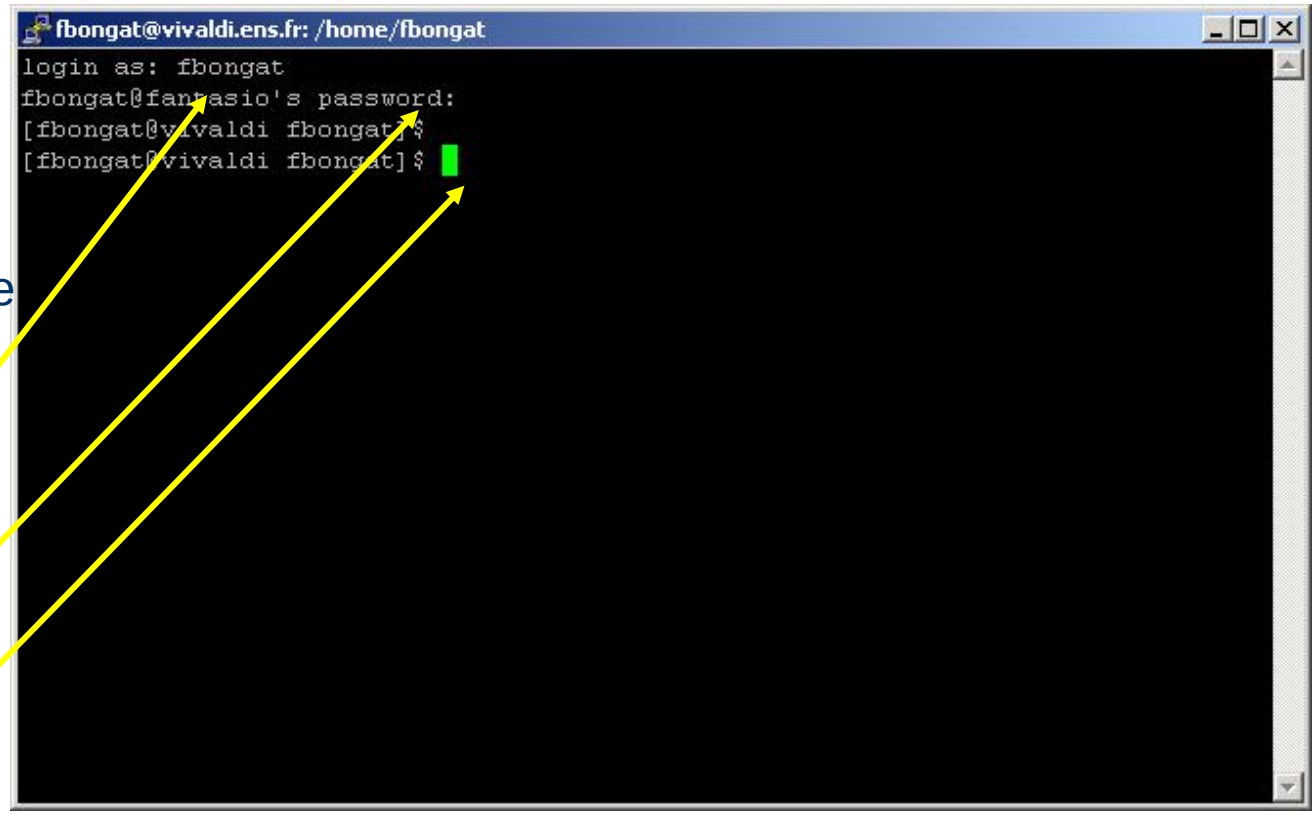


Ajout de la clé publique dans la base de registres (équivalent au fichier known_hosts)

Répondre « oui » pour passer à la suite !

Côté client : PuTTY - ssh

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

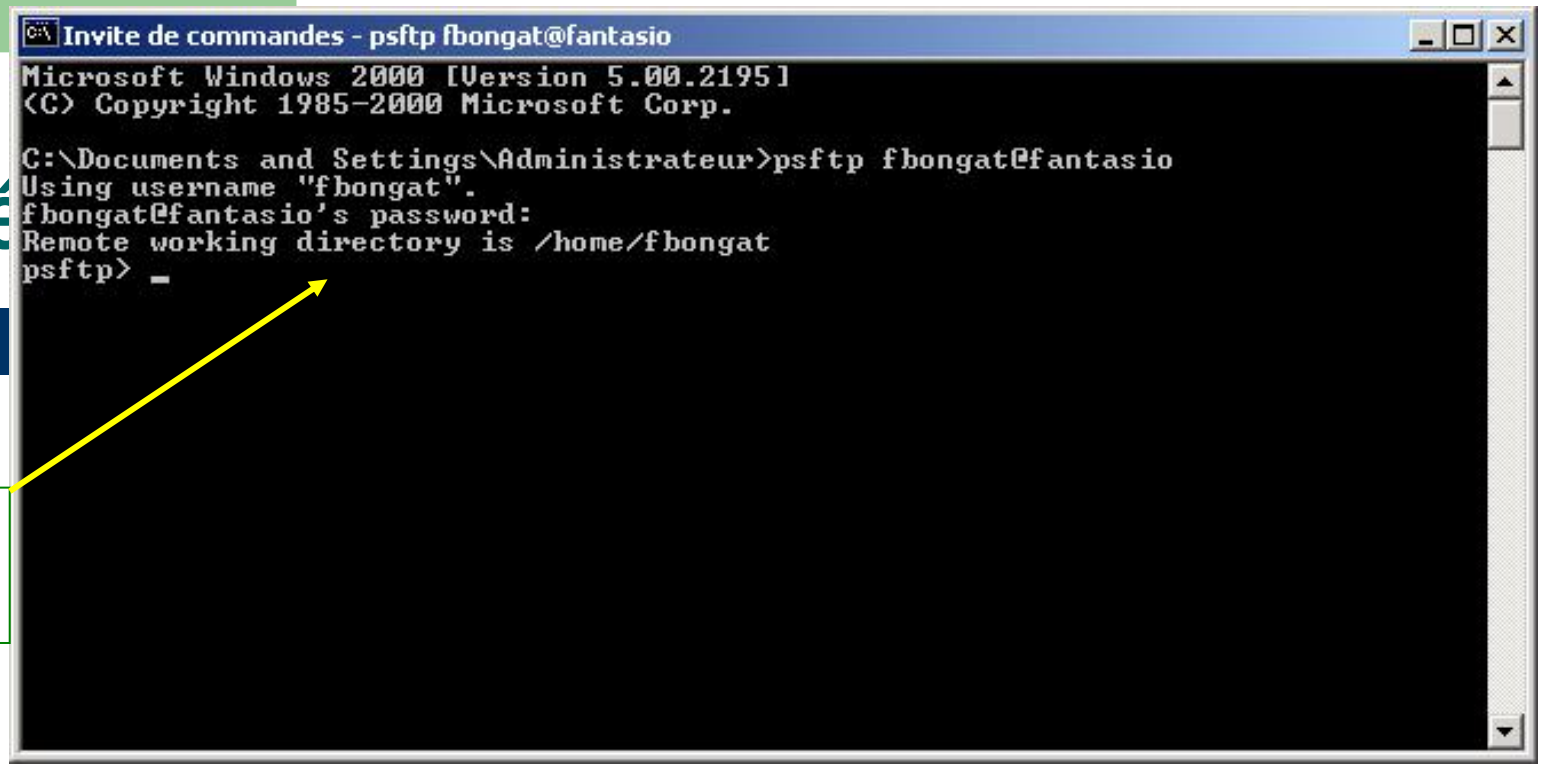


Taper le login après le prompt « *login as* »

Puis entrer le mot de passe

Connexion en ssh sur la machine distante avec PuTTY

Côté

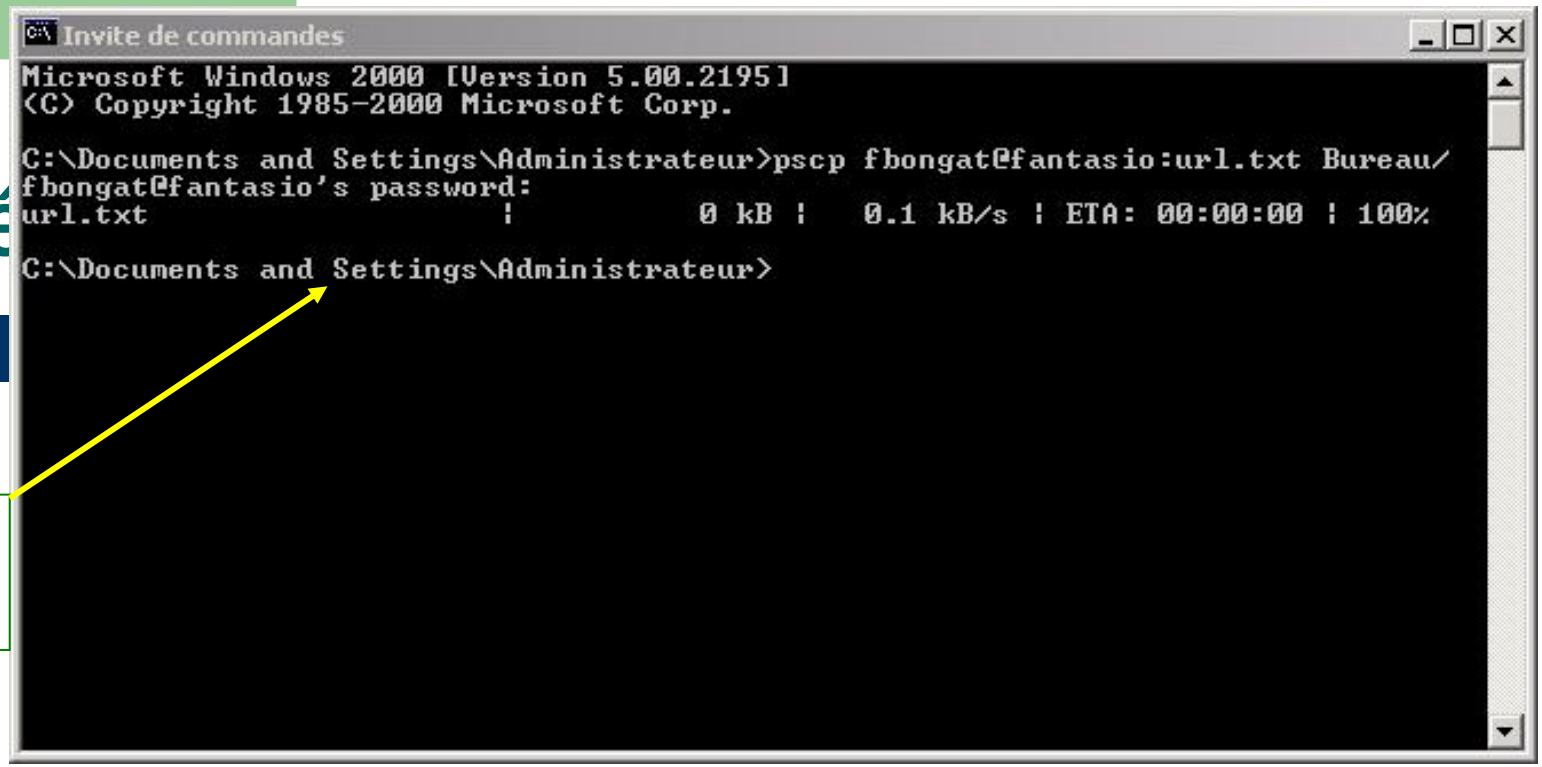


psftp fonctionne comme sftp dans un shell

- 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



pscp fonctionne comme scp dans un shell

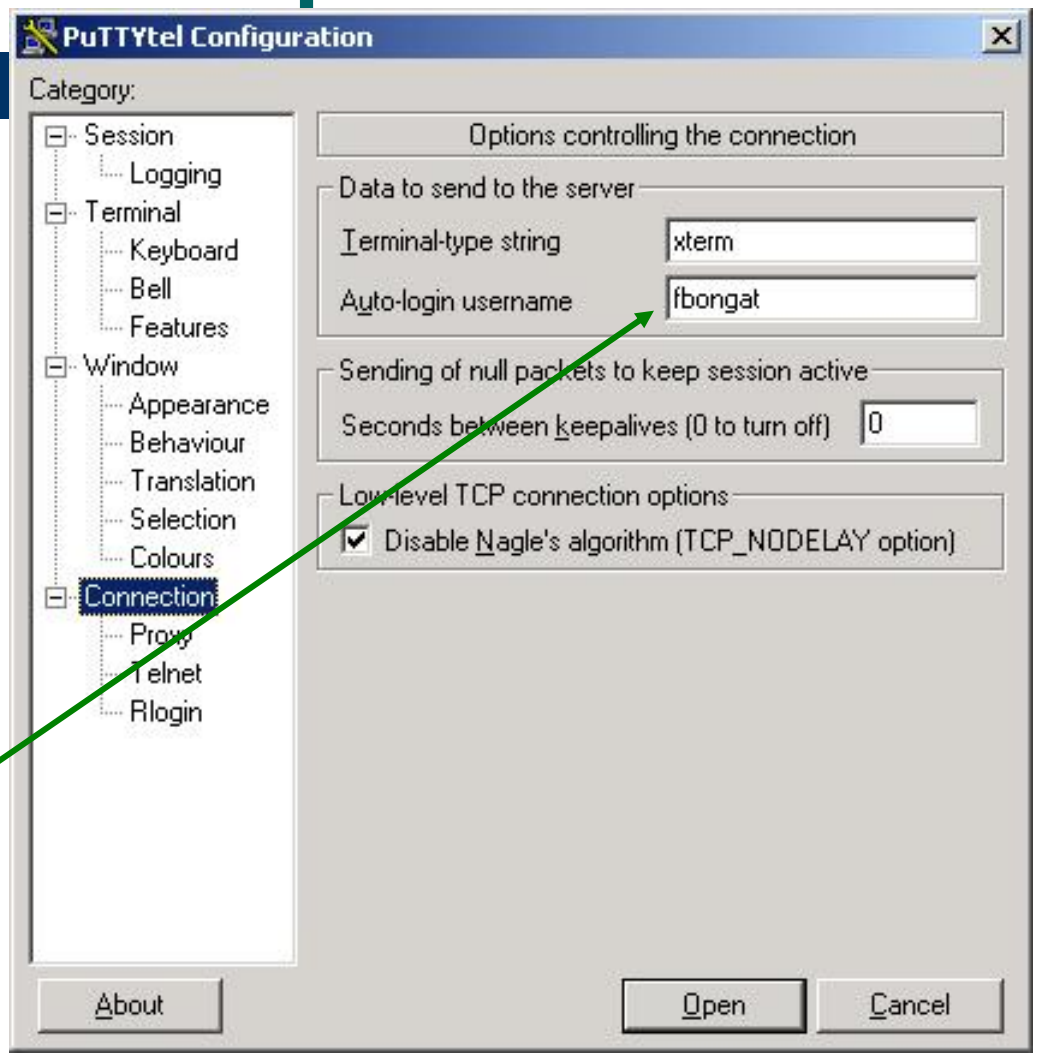
- SSH et Windows : PuTTY
  - SCP
    - Ouvrir une console « ***Invite de commandes*** »
    - Tapper ***pscp*** dans cette fenêtre

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

Côté client : PuTTY - profiles

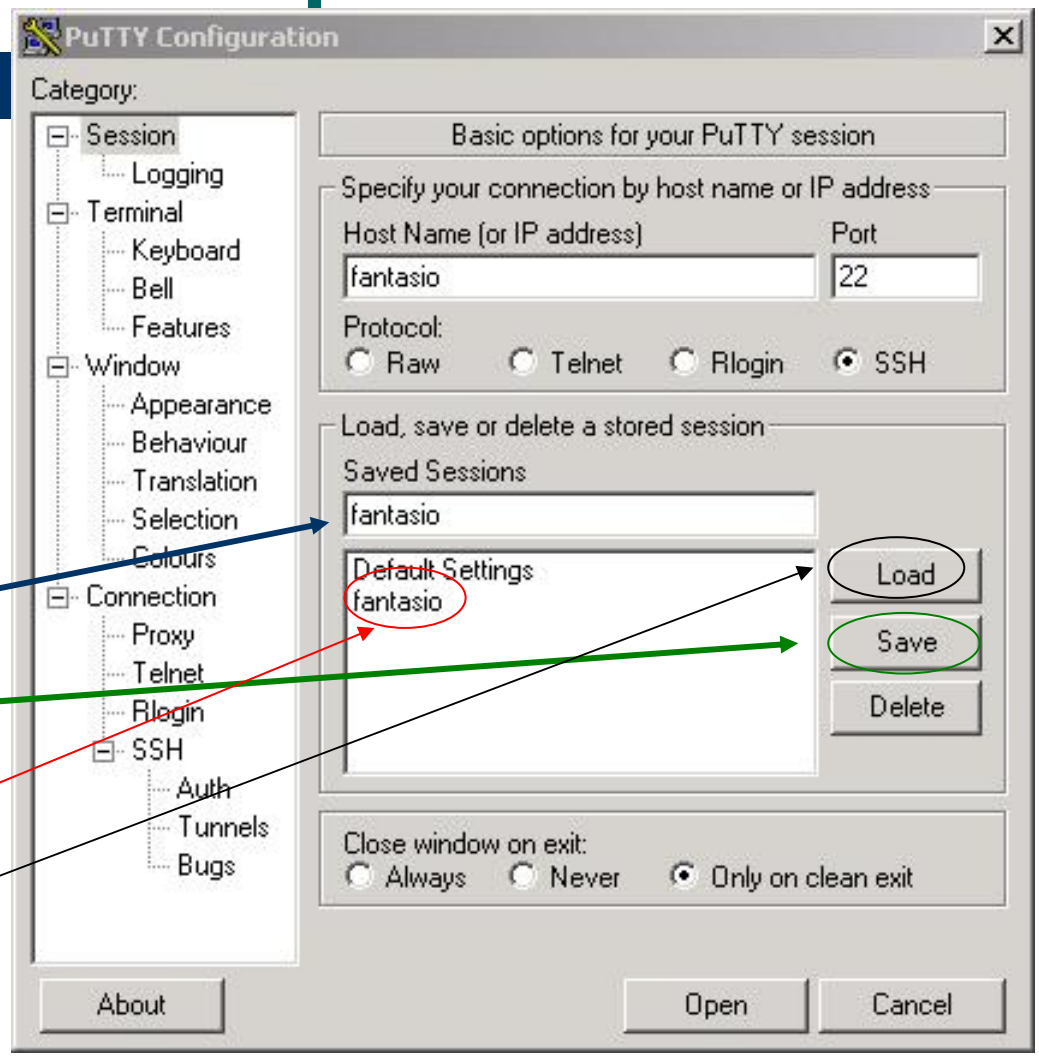
- SSH et Windows : PuTTY
  - Configuration avancée
    - Afin d'avoir à éviter de entrer le *login* dans les différentes fenêtres de commande (ssh, sftp, scp), il faut remplir le champ :

      Auto-Login username



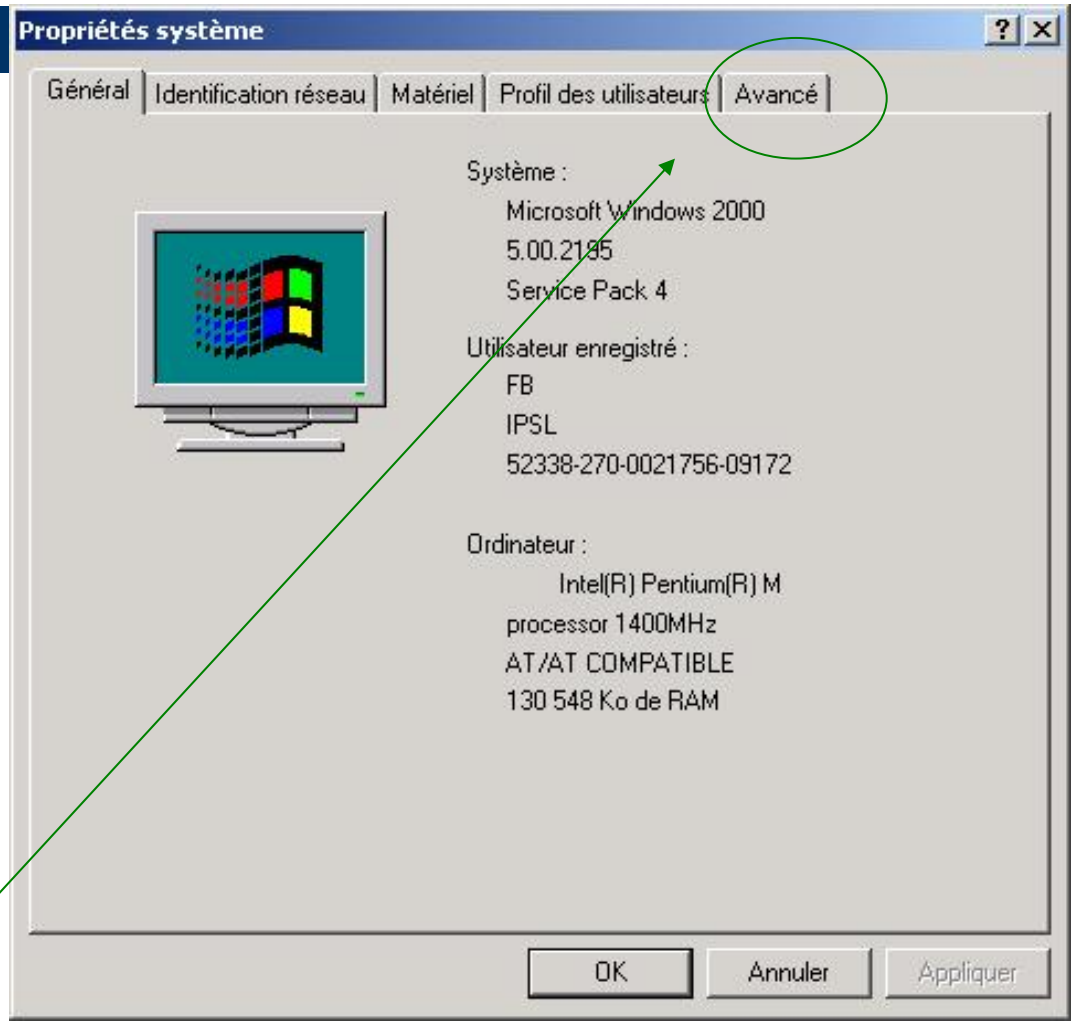
Côté client : PuTTY - profiles

- 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 sélectionner
      - Puis le charger ***load***



Côté client : PuTTY - path

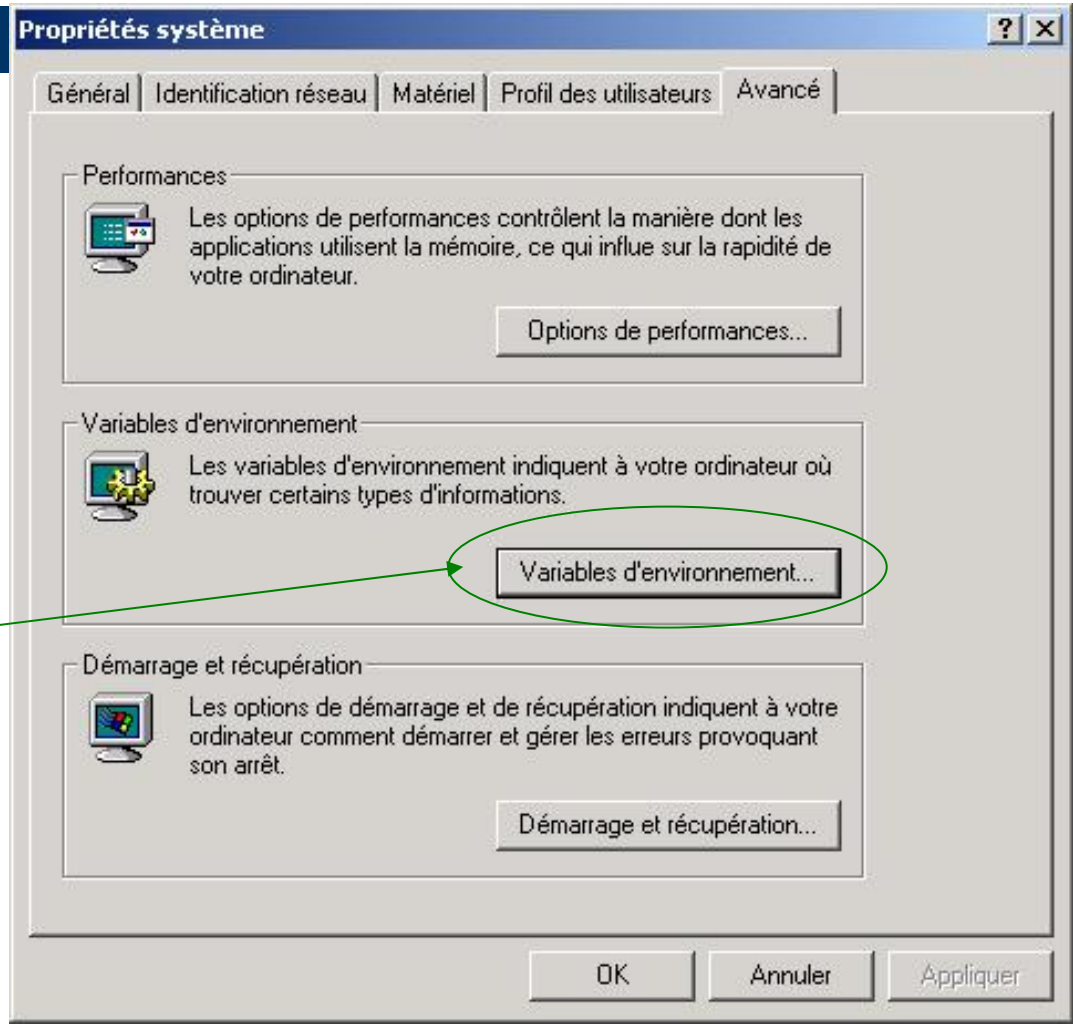
- SSH et Windows : PuTTY
  - Ajout du *path*
    - afin de trouver les binaires psftp, pscp, putty dans les fenêtres de console, il est nécessaire d'ajouter le path dans les paramètres système de Windows
    - Ouvrir le « ***Panneau de configuration*** »
    - Double cliquer sur « ***Système*** »
    - La fenêtre suivante apparaît



Cliquer sur l'onglet « Avancé »

Côté client : PuTTY - path

- SSH et Windows : PuTTY
  - Ajout du *path*
    - afin de trouver les binaires psftp, pscp, putty dans les fenêtres de console

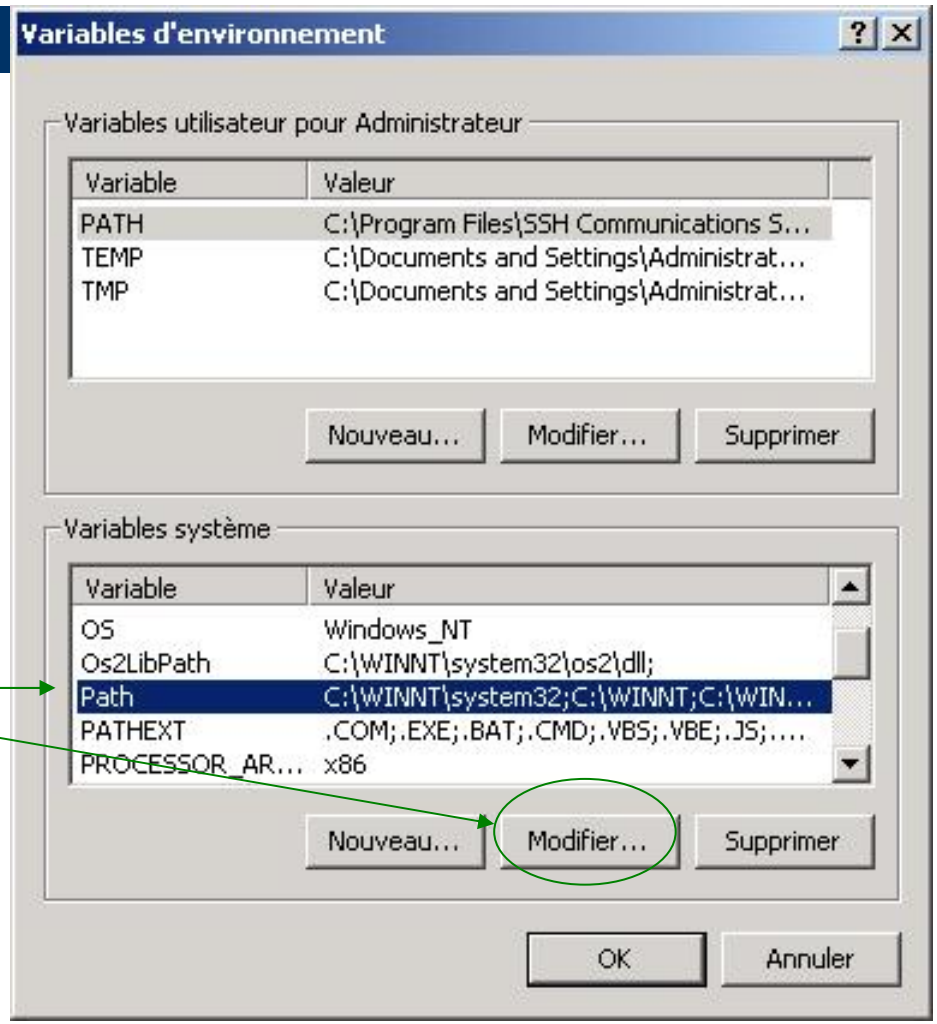


Cliquer sur « Variables d'environnement »

Côté client : PuTTY - path

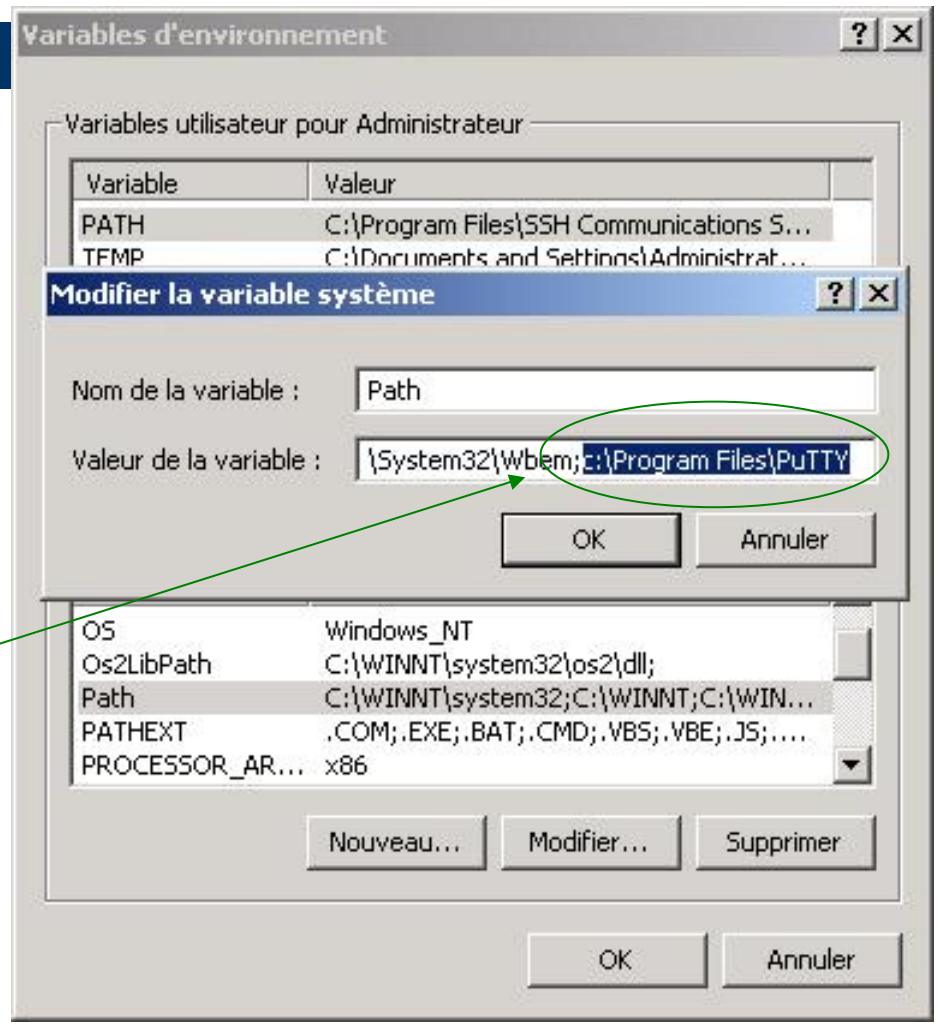
- SSH et Windows : PuTTY
  - Ajout du *path*
    - afin de trouver les binaires psftp, pscp, putty dans les fenêtres de console

Cliquer sur « Modifier » de la variable *Path* des « Variables système »



Côté client : PuTTY - path

- SSH et Windows : PuTTY
  - Ajout du *path* afin de trouver les binaires psftp, pscp, putty dans les fenêtres de console

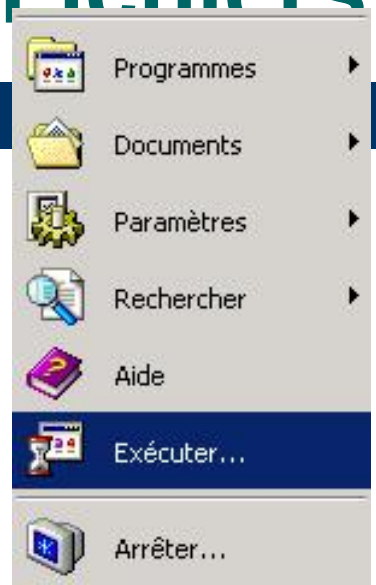


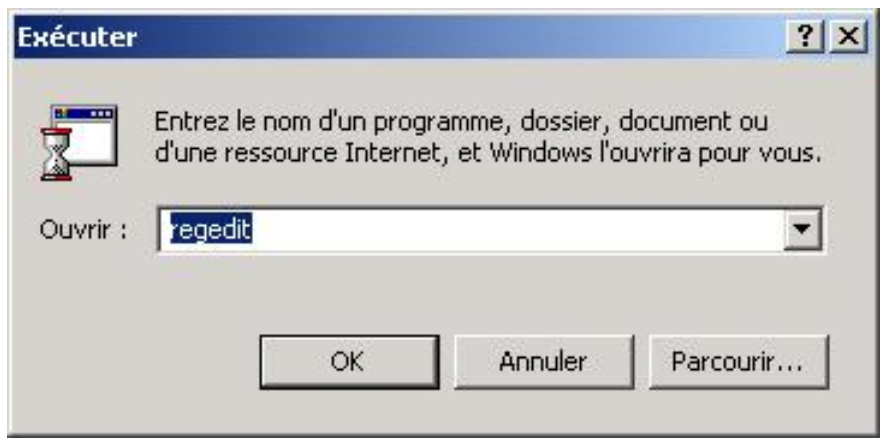
Ajouter le path « c:\Program Files\PuTTY » dans la valeur de la variable *path* et valider

la séparation de 2 path est « ; »

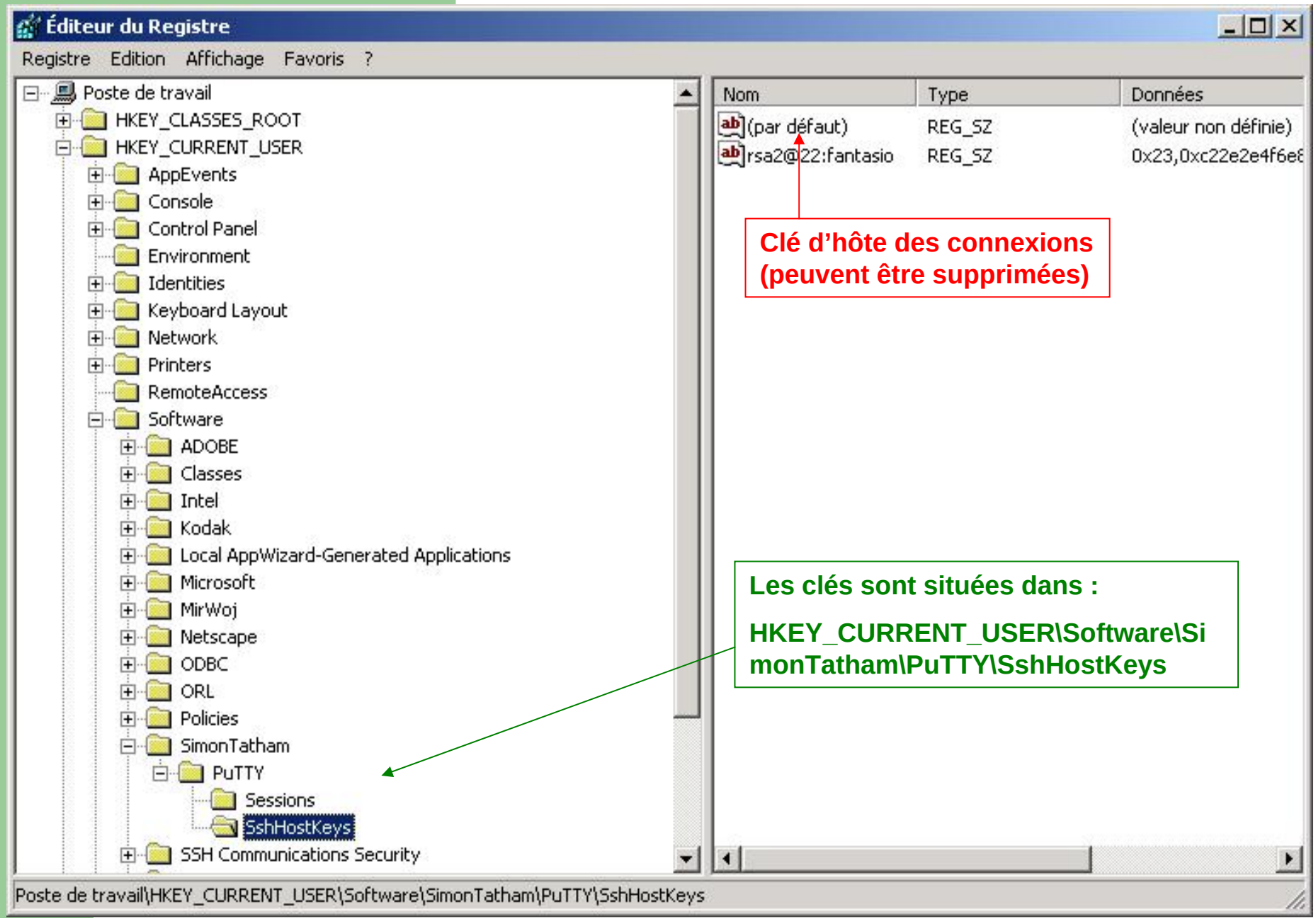
côté client : PuTTY – Fichiers

- SSH et Windows : PuTTY
  - Fichiers des clés d'hôtes
    - Ces clés sont insérées dans la base de registres de Windows.
    - Pour en supprimer une, lancer le programme « **regedit** »
    - Via le menu *Exécuter* du menu *Démarrer*
    - Chaque utilisateur peut gérer ses clés via la base de registres (ce n'est pas réservé à l'administrateur) :
      - HKEY_CURRENT_USER



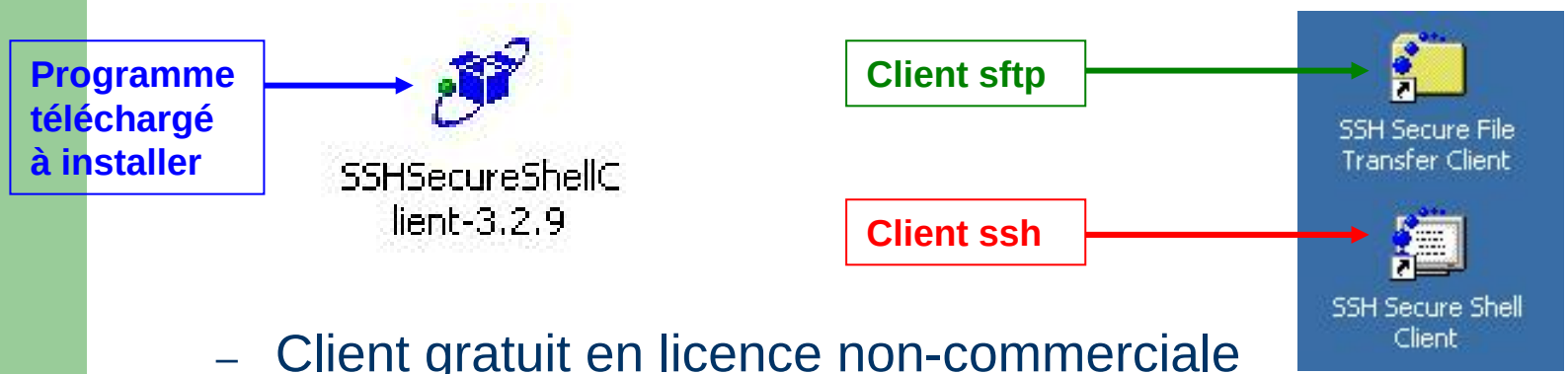


SSH et Windows : PuTTY : **Fichier des clés d'hôtes**



côté client : ssh.com

- SSH et Windows : ssh.com (Ssh Tectia)



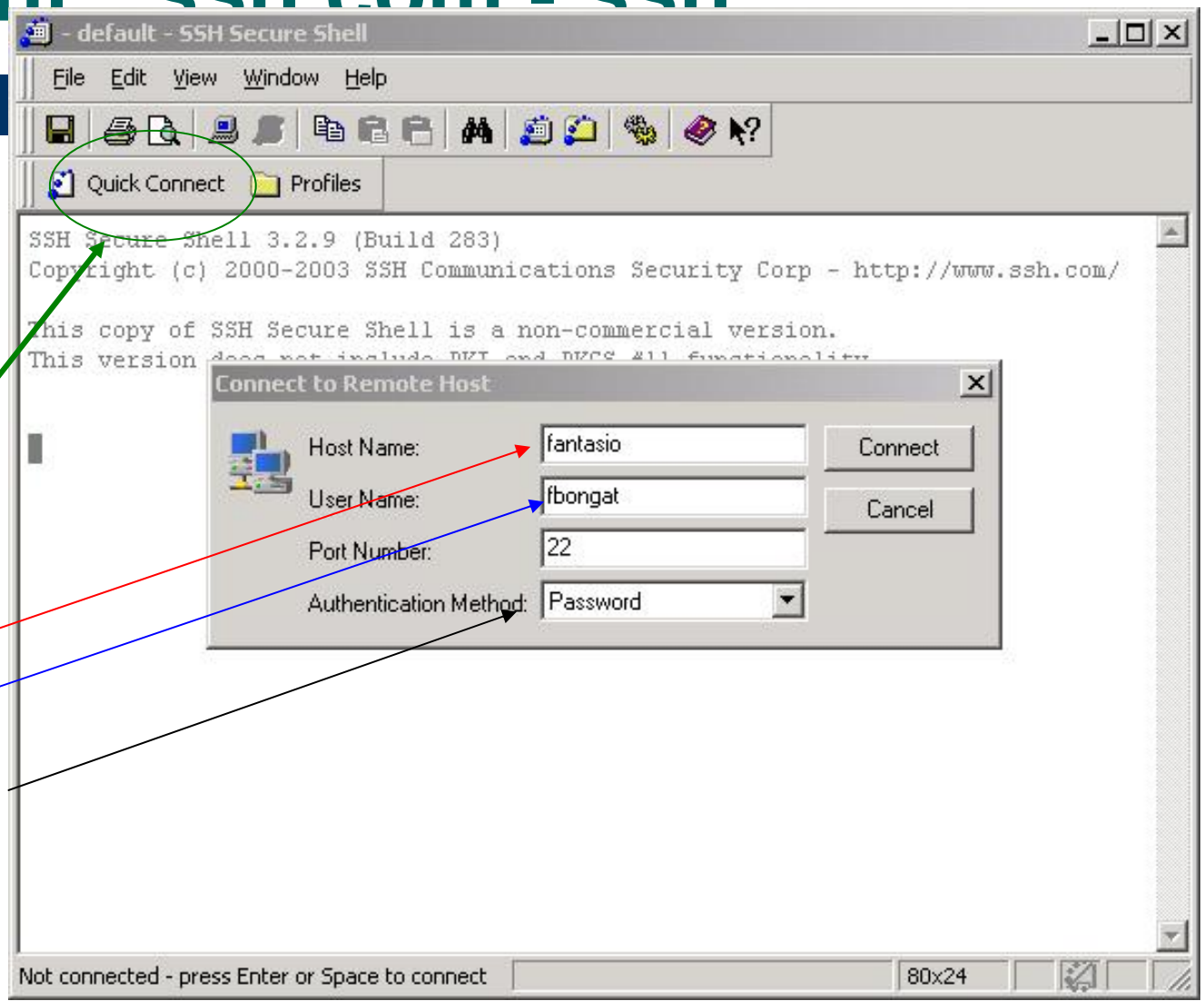
  - Client gratuit en licence non-commerciale
  - Boîte à outils limitée à un client de connexion ***ssh*** et à un client ***sftp***
  - Ergonomie très agréable, notamment pour le transfert de données

connexion par défaut en ***ssh***

côté client : ssh.com - ssh

- SSH et Windows : ssh.com (Ssh Tectia)

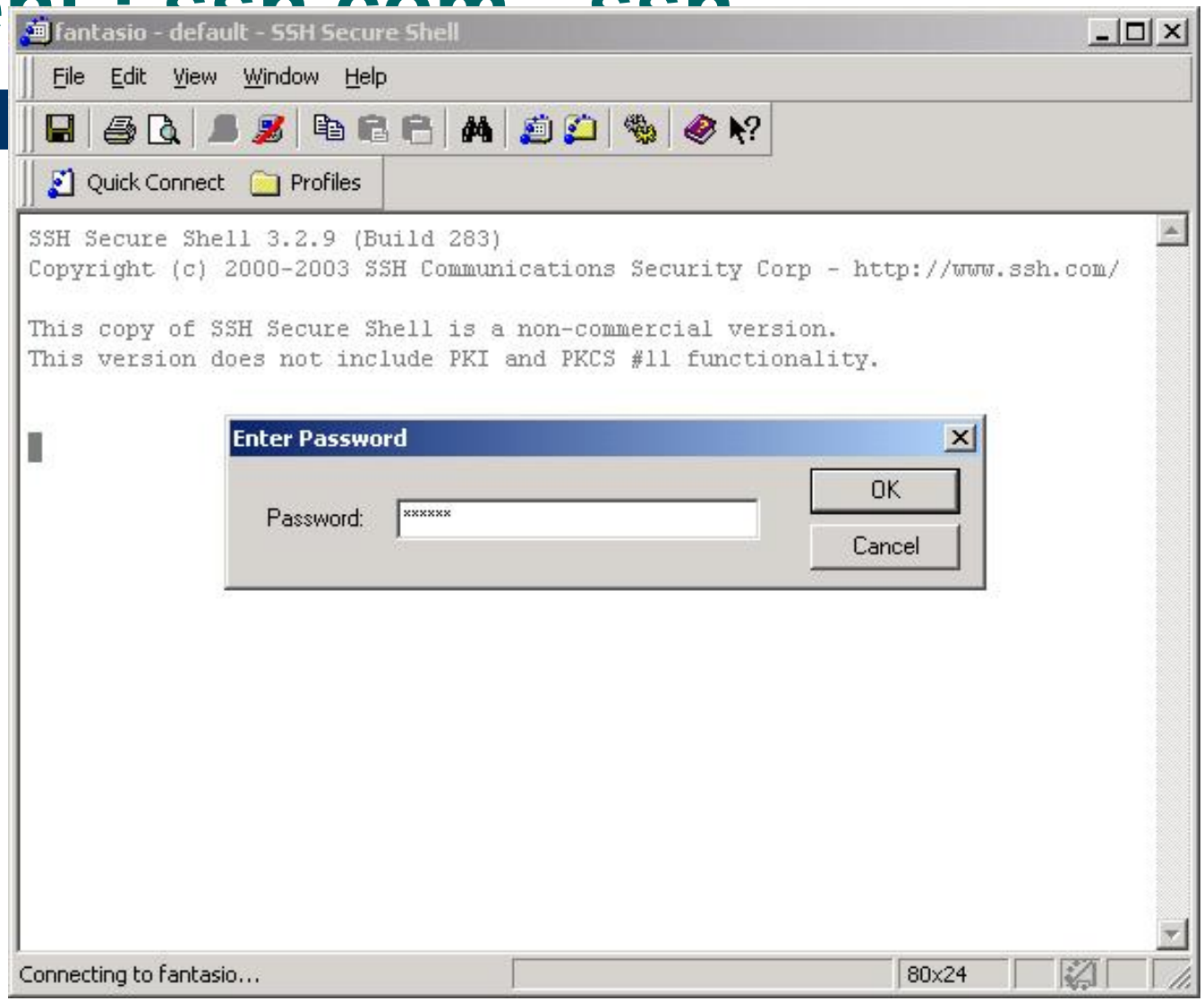
  - Cliquer sur l'icône du client ssh
  - Puis sur :
  - ***Quick Connect***
  - Remplir la fenêtre qui apparaît
    - Nom de l'hôte distant
    - Login
    - Type de méthode d'accès : par mot de passe



connexion par défaut en ***ssh***

côté client : ssh.com - ssh

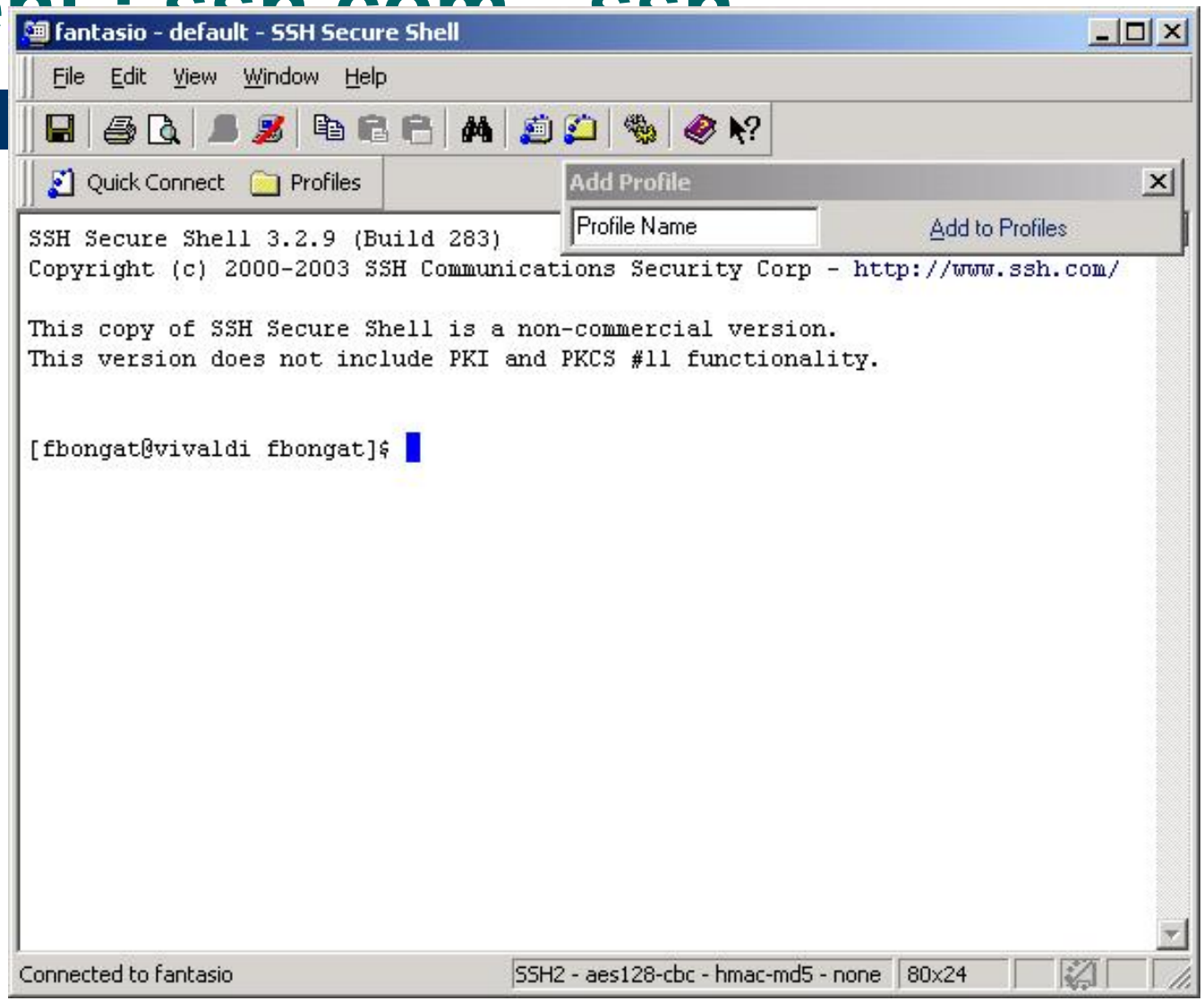
- SSH et Windows : ssh.com (Ssh Tectia)
  - Attendre la demande du mot de passe dans une boîte de dialogue qui va apparaître, puis entrer le mot de passe



connexion par défaut en ***ssh***

côté client : ssh.com - ssh

- SSH et Windows : ssh.com (Ssh Tectia)
  - Fenêtre de connexion ssh

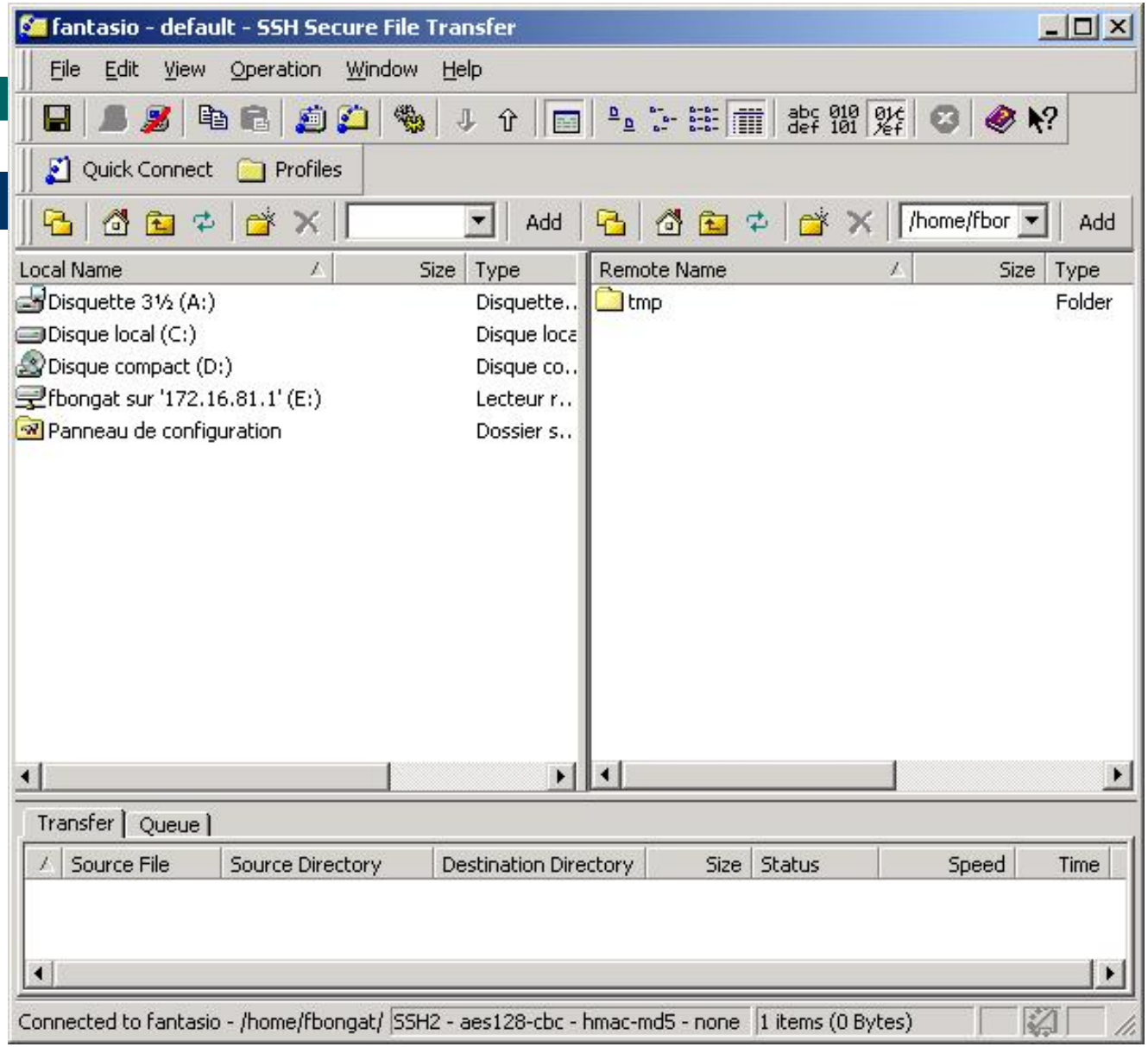


côté clien

connexion par défaut en ***sftp***

- SSH et Windows : ssh.com (Ssh Tectia)

  - Cliquer sur l'icône du client sftp
  - Puis sur :
    - ***Quick Connect***
    - Remplir la fenêtre qui apparaît
      - Nom de l'hôte distant
      - Login
      - Type de méthode d'accès : par mot de passe
    - Puis attendre et entrer le mot de passe (voir ssh)

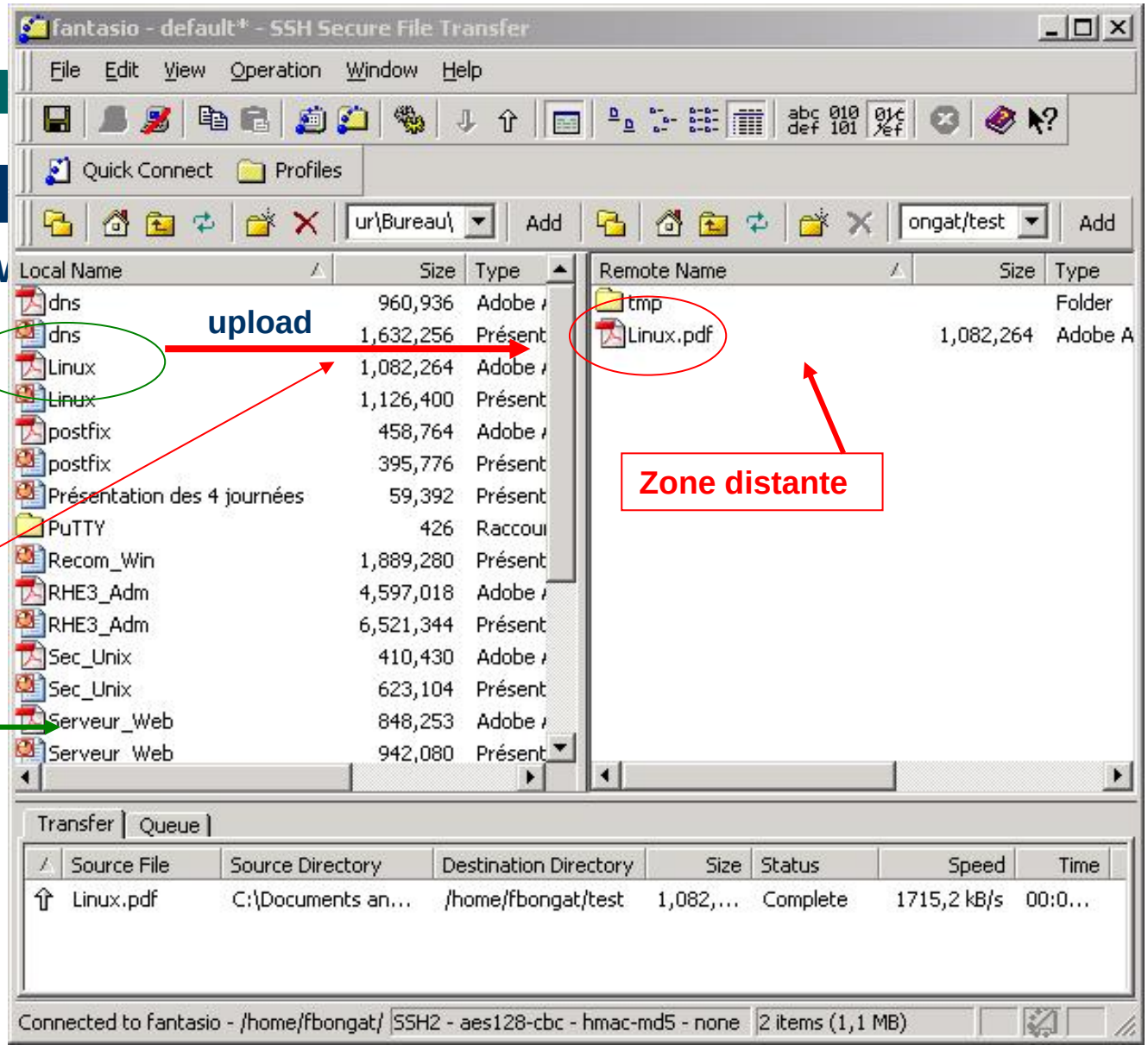


connexion par défaut en ***sftp***

côté clien

- **SSH et Windov (Ssh Tectia)**

- Exemple d'un transfert du fichier *Linux.pdf* vers le site distant



Téléchargement par drag & drop

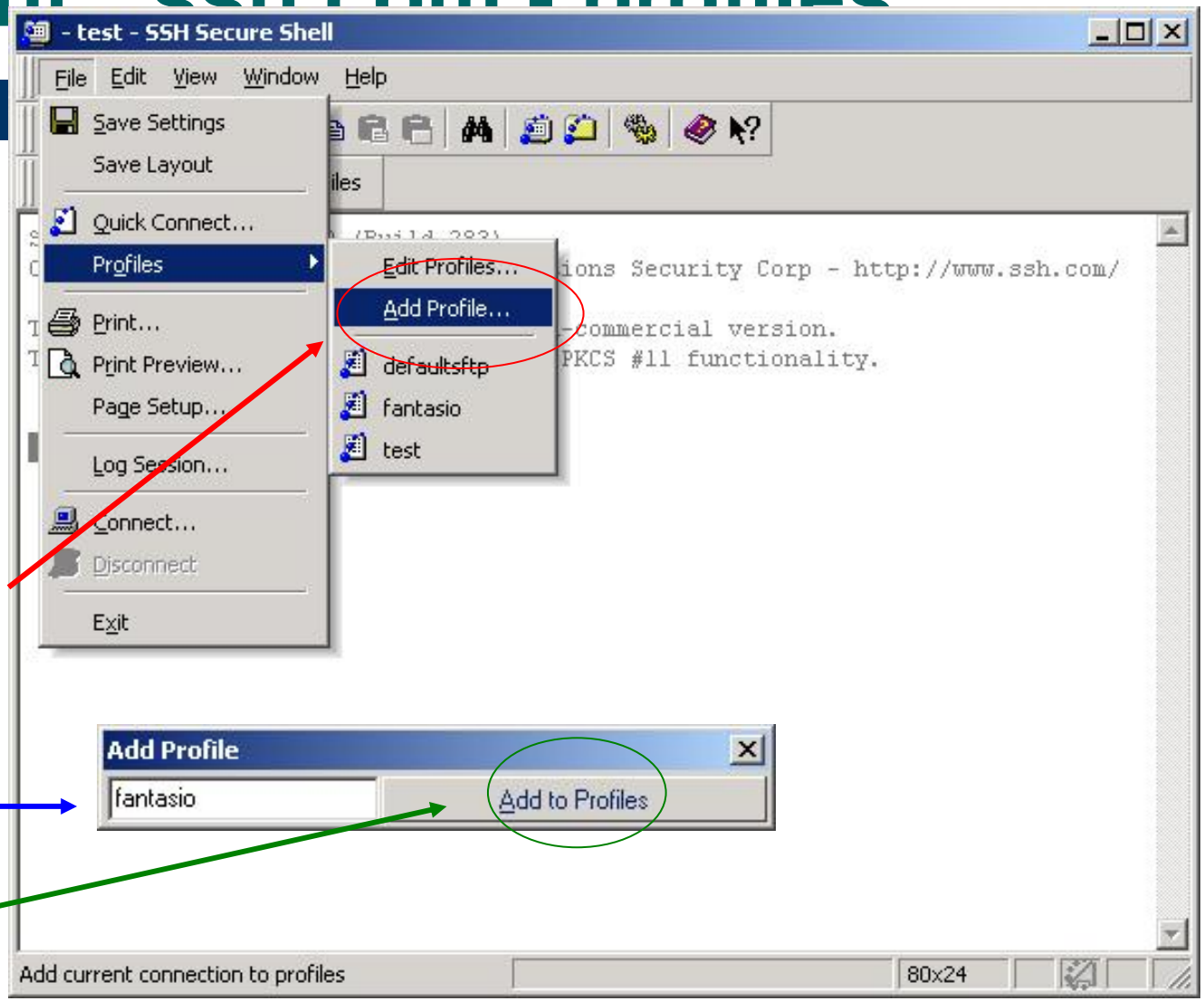
Zone locale Windows

Résumé (ou état) du téléchargement

Configuration avancée : *ajouter un profile utilisateur*

côté client : ssh.com - profiles

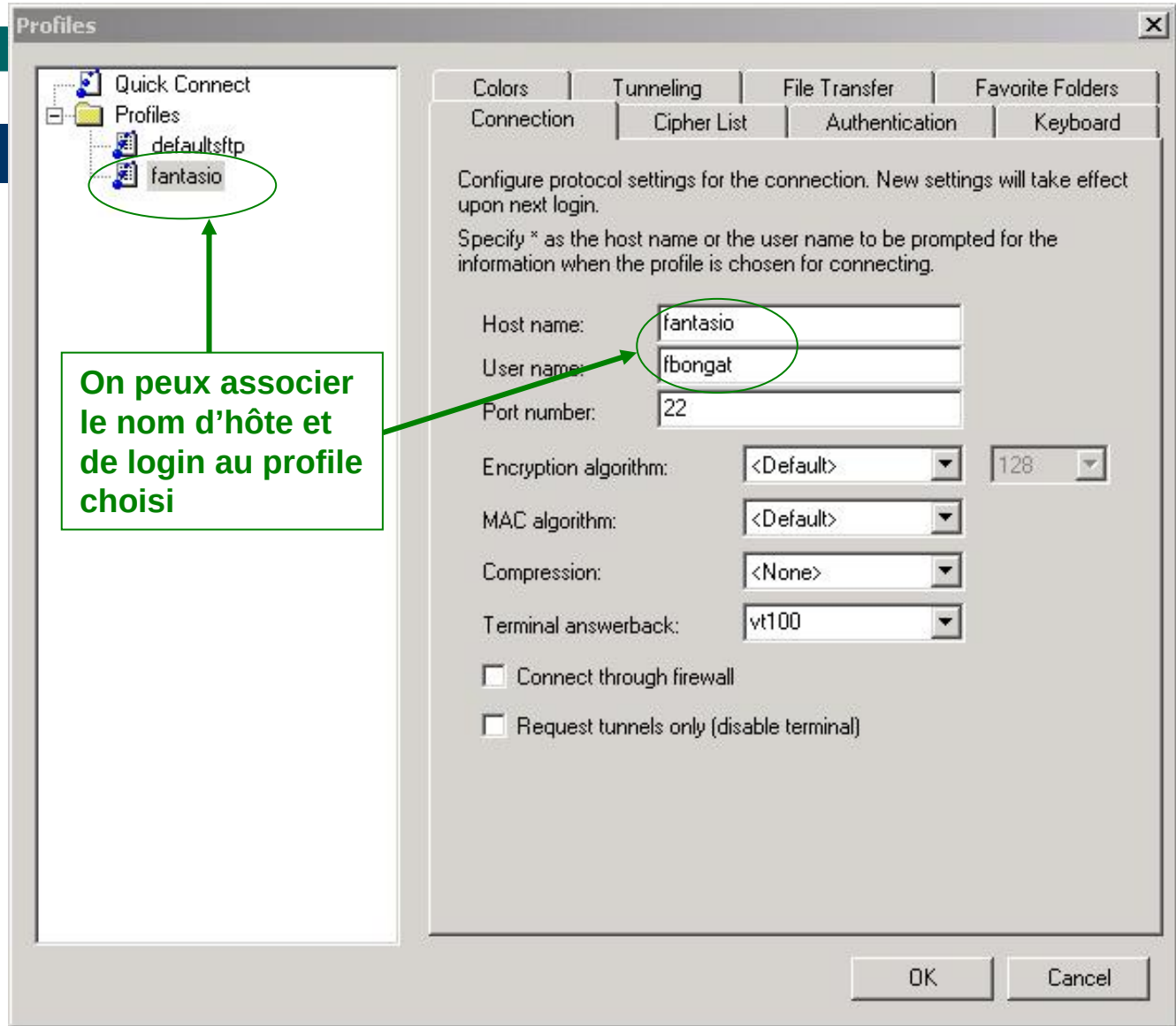
- SSH et Windows : ssh.com (Ssh Tectia)
  - Cliquer sur l'icône du client ***ssh***
  - Puis aller dans le menu ***File***
  - Puis sélectionner ***Profiles***
  - Choisir ***Add Profile***
  - Entrer le nom du profile dans la boîte de dialogue qui s'ouvre ensuite :
  - Valider sur ***Add to Profiles***



côté clien

Configuration avancée : *éditer un profile utilisateur*

- SSH et Windows : ssh.com (Ssh Tectia)
- Permet de paramétrer ces options attachées à un profile
  - Cliquer sur l'icône du client ***ssh***
  - Puis aller dans le menu ***File***
  - Puis sélectionner ***Profiles***
  - Choisir ***Add Profile***

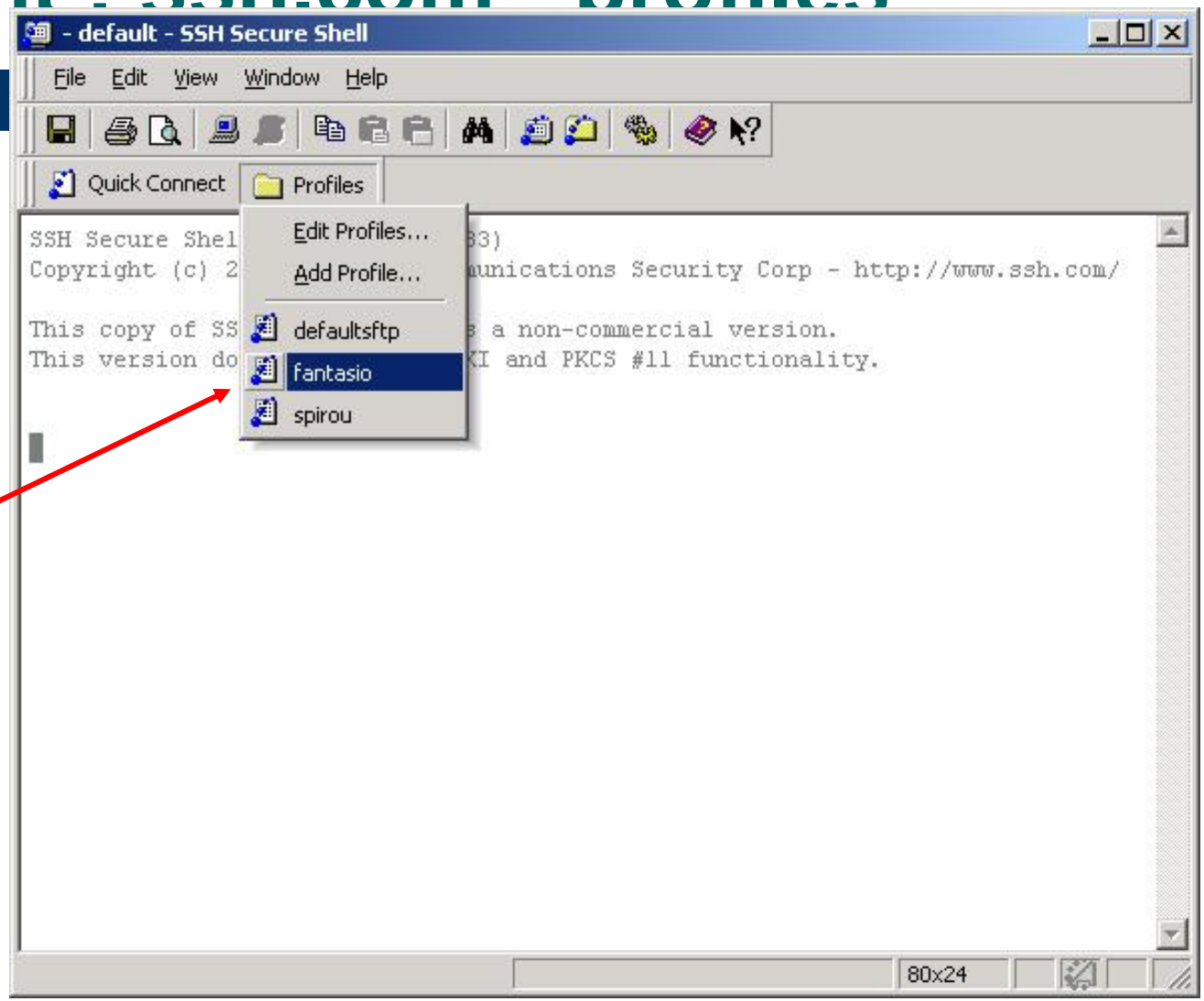


côté client : ssh.com - profiles

Configuration avancée : *lancement d'un profile*

- SSH et Windows : ssh.com (Ssh Tectia)

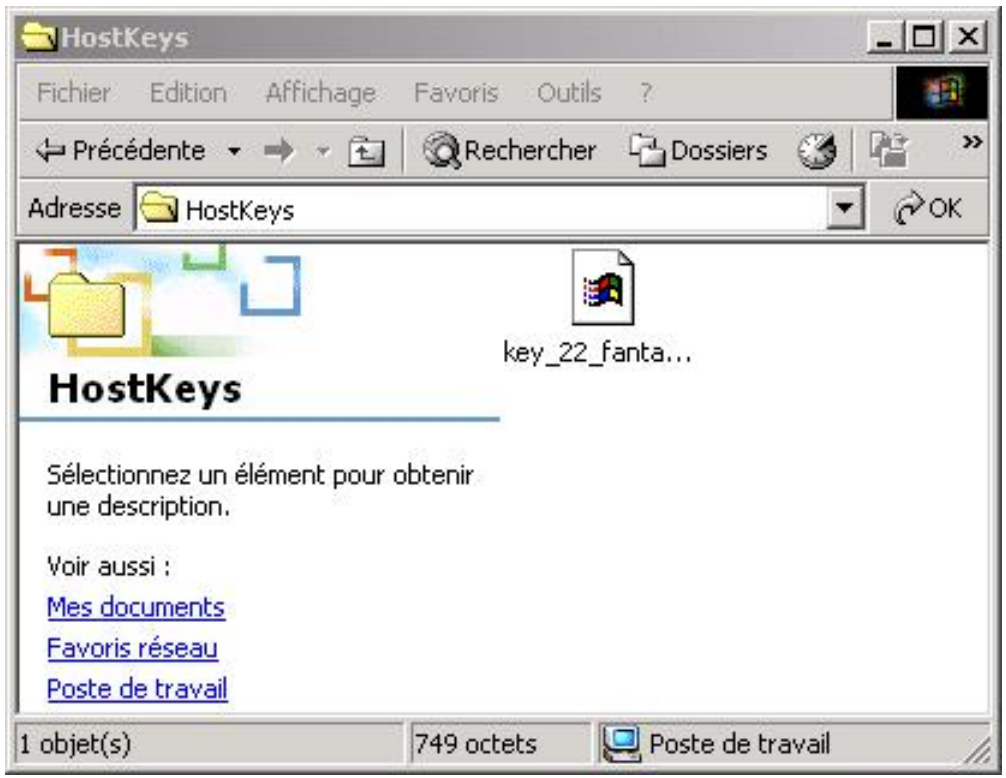
Lancement de la connexion à partir d'un profile pré-configuré :



côté client : ssh.com - fichiers

- SSH et Windows : ssh.com (Ssh Tectia)
  - Les clés publiques des hôtes se trouvent dans le dossier ***HostKeys*** dans Documents and Settings de chaque utilisateur

Configuration avancée : *ajouter un profile utilisateur*



côté client : ssh et X11

- X11 Forwarding :
  - 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
  - Sous OpenSSH, serveurs et clients Unix sont configurés ***par défaut*** pour transmettre des données X11 dans le canal sécurisé ( **!** pas sur MacOS X et FreeBSD 5.x, il faut le configurer)
  - En cas de soucis, un test de fonctionnement peut être effectué en initiant la connexion avec l'option ***-X*** (active l'X11 en cas de non configuration dans le fichier /etc/ssh/ssh_config : voir côté serveur):

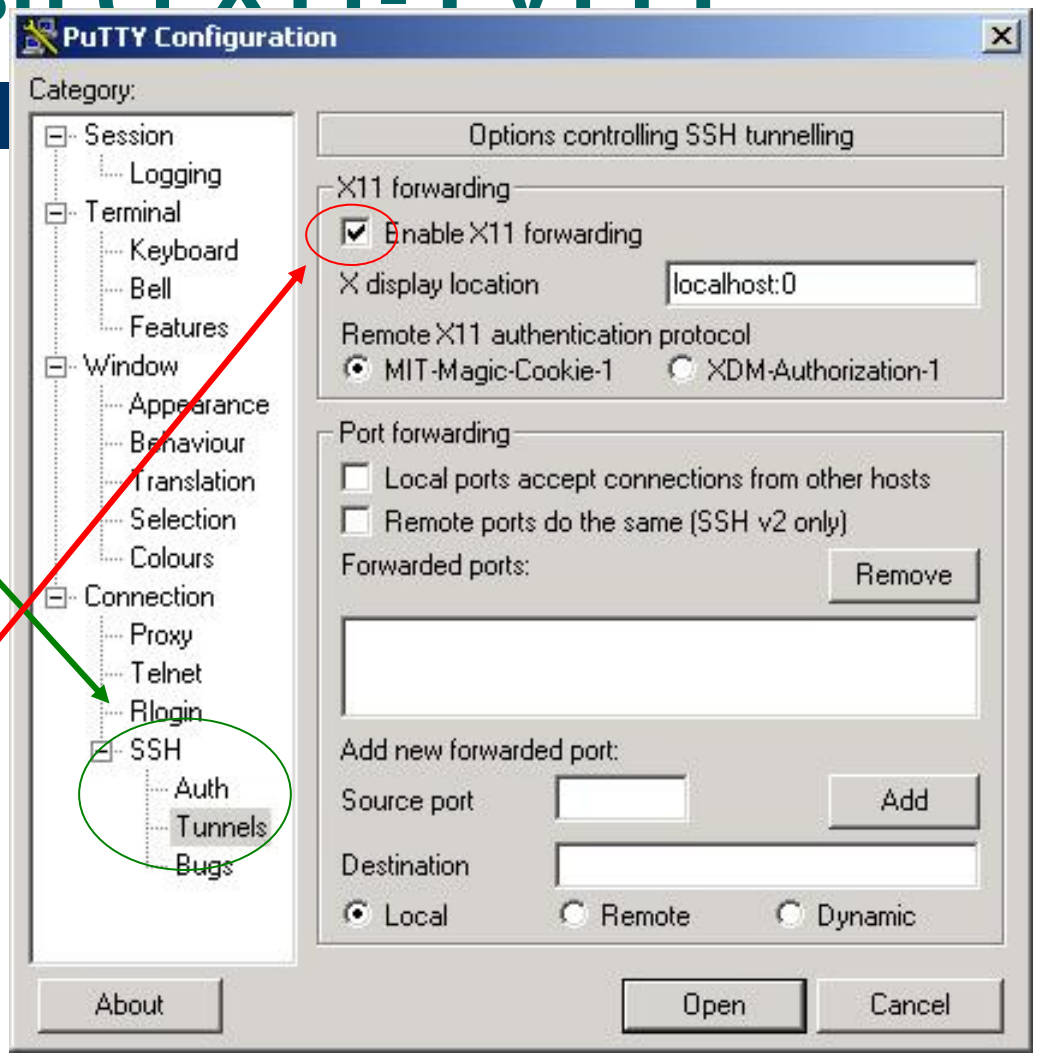
    ssh –X login@machine

côté client : ssh et X11- PuTTY

- X11 Forwarding :
  - Client PuTTY
    - Lancer ***putty***

    - Menu ***SSH → Tunnels***
    - Cocher la case :

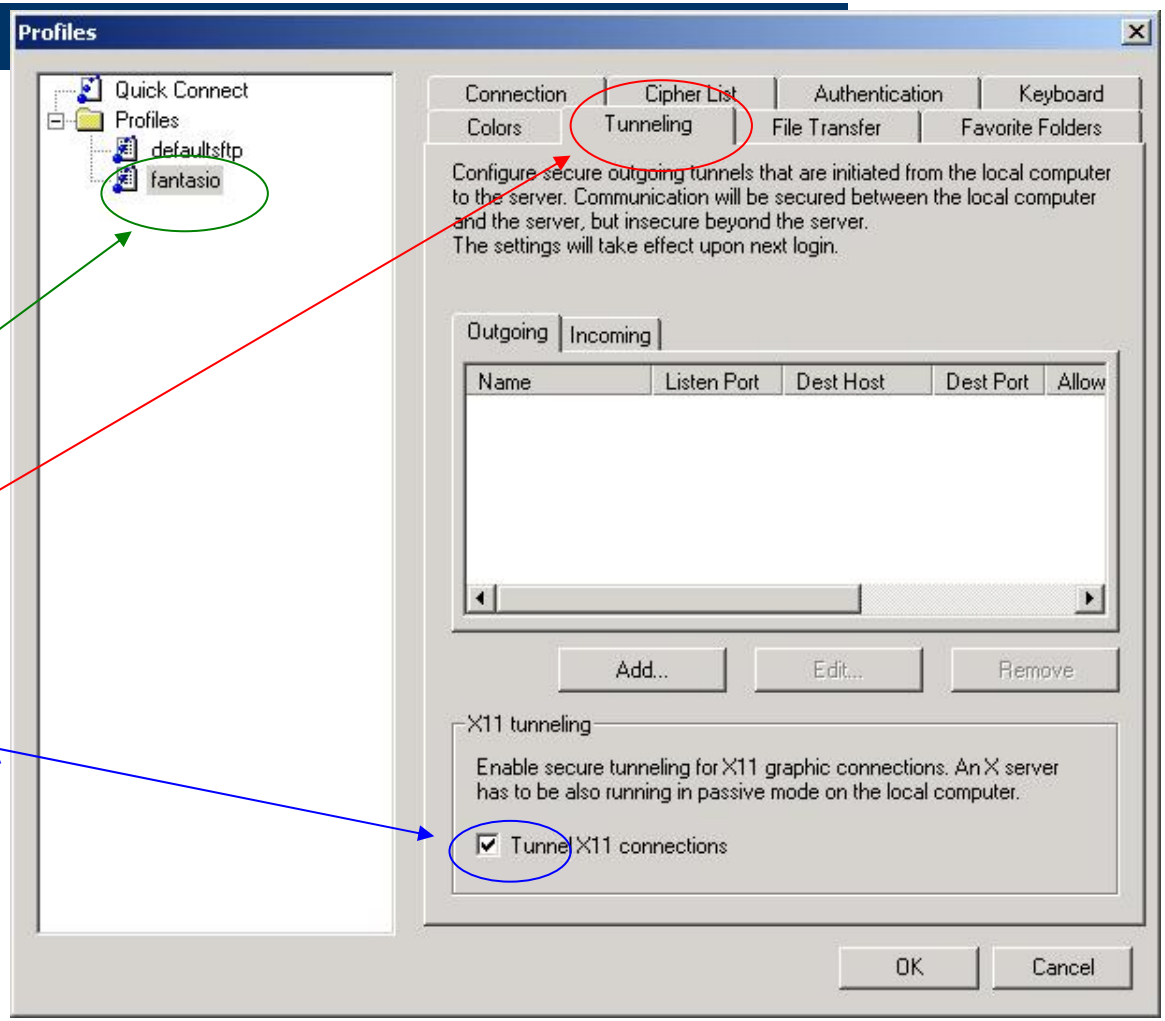
      Enable X11 forwarding
  - N'oubliez pas sous Windows, il faut aussi lancer un émulateur X11



côté client : ssh et X11 – ssh.com

- X11 Forwarding :
  - Client ssh.com (Tectia)

    - Lancer ***ssh***
    - Menu ***File*** → ***Profiles***
    - Ouvrir le profil désiré via : ***Edit Profiles***
    - Sélectionner l'onglet : ***Tunneling***
    - Cocher la case : ***Tunnel X11connections***
  - N'oubliez pas sous Windows, il faut aussi lancer un émulateur X11



côté client : autres services

- Des services additionnels :
  - Décrits dans la suite :
    - Agent SSH : utilisation en mode authentification forte. Programme qui garde les clés privées en mémoire et qui fournit les services d'authentification aux autres clients SSH.
    - Agent Forwarding : utilisation en mode authentification forte. Relaye les demandes d'authentification entre les différents clients-serveurs jusqu'au client initial (agent de mandatement)
    - Tunneling : permet de rediriger tout service TCP à travers un canal chiffré.

Côté serveur

- SSH vu du côté serveur …

Côté serveur : sshd

- Le serveur : sshd
  - Répertoire du serveur : /etc/ssh
    - Deux fichiers de configuration :
      - sshd_config , ssh_config
    - 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 : fichiers

- 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 s

sshd_config

(partie 1)

Fichier Édition Affichage Terminal Aller à Aide

```
#Port 22
Protocol 2
#ListenAddress 0.0.0.0
#ListenAddress ::

# HostKey for protocol version 1
HostKey /etc/ssh/ssh_host_key
# HostKeys for protocol version 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
#KeyRegenerationInterval 3600
#ServerKeyBits 768

# Logging
#obsoletes QuietMode and FascistLogging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 120
PermitRootLogin yes
#StrictModes yes

#RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      .ssh/authorized_keys

# rhosts authentication should not be used
RhostsAuthentication no
# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts no
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#RhostsRSAAuthentication yes
# similar for protocol version 2
HostbasedAuthentication yes
# Change to yes if you don't trust ~/.ssh/known_hosts for
# RhostsRSAAuthentication and HostbasedAuthentication
IgnoreUserKnownHosts no
-- INSERTION --                                    53,69         23%
```

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, ne fonctionne qu'avec la première version du protocole
- **PubkeyAuthentication yes** : méthode d'authentification forte (rsa ou dsa) en V2
- **AuthorizedKeysFile .ssh/authorized_keys** : nom et localisation du fichier de clés publiques individuelles sur les hôtes locaux.

Côté

sshd_config

(partie 2)

Fichier Édition Affichage Terminal Aller à Aide

```
# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
#AFSTokenPassing no

# Kerberos TGT Passing only works with the AFS kaserver
#KerberosTgtPassing no

# Set this to 'yes' to enable PAM keyboard-interactive authentication
# Warning: enabling this may bypass the setting of 'PasswordAuthentication'
#PAMAuthenticationViaKbdInt no

X11Forwarding yes
X11DisplayOffset 10
X11UseLocalhost yes
#PrintMotd yes
#PrintLastLog yes
#KeepAlive yes
#UseLogin no
UsePrivilegeSeparation yes
#PermitUserEnvironment no
#Compression yes

#MaxStartups 10
# no default banner path
#Banner /some/path
#VerifyReverseMapping no

# override default of no subsystems
Subsystem       sftp    /usr/lib/ssh/sftp-server
```

92,62-71 Bas

Après modification de ce fichier, relancer le serveur sshd

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

- Quelques options méconnues :
  - AllowUsers login : autorise le compte local et seulement ce compte à se connecter via ssh
  - DenyUsers login : indique que ce compte ne pourra pas recevoir de connexions ssh
    - Possibilité de gérer l'hôte source : login@host
    - (AllowGroups group ou DenyGroups group)
    - Si il y a plusieurs comptes ou groupes, la syntaxe est séparation des noms par des espaces.

Côté serveur : ssh_config

ssh_config

Fichier Édition Affichage Terminal Aller à Aide

```
# Host *
#   ForwardAgent no
#   ForwardX11 no
#   RhostsAuthentication no
#   RhostsRSAAuthentication no
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   HostbasedAuthentication no
#   BatchMode no
#   CheckHostIP yes
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   Port 22
#   Protocol 2,1
#   Cipher 3des
#   Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour,aes192-cbc,aes2
56-cbc
#   EscapeChar ~

Host *
  ForwardAgent yes
  ForwardX11 yes
  Protocol 2,1
  StrictHostKeyChecking no
  HostbasedAuthentication yes
                                                          43,1          Bas
```

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)
- StrictHostKeyChecking 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)

Côté serveur : sécurité

- StrictHostKeyChecking 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 74) à l'utilisateur

- Renforcer la sécurité :
  - 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 : sécurité

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 : 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//jRCLUgV7XD18xFB79LLOQq/hz+f
sHON8Mr9tuiYCr1aIY45O9hyOHnejCg8PEQA17Wr9Armoju7CMug/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 : clés du serveur

- Exemples :
  - Clé publique (rsa) d'un serveur

```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAxIZKEvYKJ1XGB/K7qdpDmmiMaZMi7x57g6OHPQFa
wv/wtwx4X46kpYpl1HcN92ODrW8aMdqq3d1/GUIQt1BahGMufRJLrbcg7J1CDs9+kb26t5tkoTMs
nQFwZrs7YoWAvc3fN3hMMQRS6QIHu8fOUs8Vp66bu6LmooL4jMWZ/Qk=
```

  - Clé publique (v1) d'un serveur

```
1024 35 119830301822491151119746880597259612817704675054341497287848921959058113285893940138163779751952807510020595203377997581183295978527466355715285286942652768544163906397810693955119136158422830685414937434881696605258817266360274852318438938467202776617577597321017870512950638678729325195416990770102512085491
```

Côté serveur : démarrage/arrêt

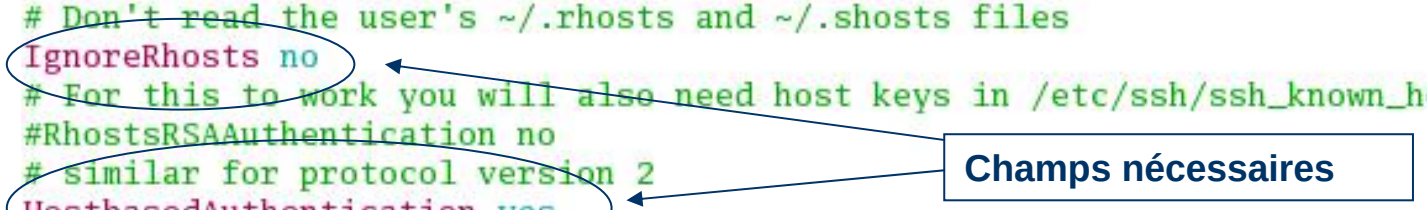
- Lancement du serveur :
  # /etc/init.d/sshd start
  ou
  # service sshd start
- Arrêt du serveur :
  # /etc/init.d/sshd stop
  ou
  # service sshd stop
- Relancer le serveur après une modification de la configuration (sshd_config) :
  # /etc/init.d/sshd restart
  ou
  # service sshd restart

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

Côté serveur : sans mot de passe

- Connexions sans mot de passe (*.shosts*) :
  - **exemple** : connexion de **spip** vers **spirou** sans mot de passe
  - Configuration du serveur distant (spirou):
    - **/etc/ssh/sshd_config**
      - **IgnoreRhosts** : désactiver cette fonction (ignorer le système équivalent à rhost)
      - **HostbasedAuthentication**: authentification par relation d'hôte de confiance (variable OpenSSH v2); à activer

```
# rhosts authentication should not be used
#RhostsAuthentication no
# Don't read the user's ~/.rhosts and ~/.shosts files
IgnoreRhosts no
# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#RhostsRSAAuthentication no
# similar for protocol version 2
HostbasedAuthentication yes
# Change to yes if you don't trust ~/.ssh/known_hosts for
# RhostsRSAAuthentication and HostbasedAuthentication
#IgnoreUserKnownHosts no
```



Côté serveur : sans mot de passe

- Connexions sans mot de passe (*.shosts*) :
  - Configuration du serveur distant (ici spirou) :
    - **/etc/ssh/shosts.equiv**
      - **Noms et adresse ip** de la machine source (ici spip)

```
[root@spirou ssh]# cat shosts.equiv
spip
spip.bdnet
172.16.158.20
[root@spirou ssh]# _
```

      - **Les droits du fichier shosts.equiv (root uniquement) :**

        chmod 600 shosts.equiv

    - **/etc/ssh/ssh_known_hosts**
      - **Mettre la clé publique de la machine source (ici spip) dans ce fichier**

Donner tous ces noms et ip

```
[root@spirou ssh]# cat ssh_known_hosts
spip,spip.bdnet,172.16.158.20 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAw2TR9mXxNludjK
rIQN4RX2NkAUBlOM716GeEmkTUgP3dqj4+2Klg+p15soJseACoZIqnDAZizmiLvnePtd3Gp3wDp4GqZ+
YsrAjMMzTKDEWJh+KzF+4ZuNXAlBgjSvgLOjRYWLJz1EvOxPZ6KCfovpnrDp+O2TUIeAIjxl/23iM=
```

Côté serveur : sans mot de passe

- Connexions sans mot de passe (*.shosts*) :
  - Configuration du serveur source (ici spip):
    - Fichier de configuration du client local ssh
    - **/etc/ssh/ssh_config**
      - **EnableSSHKeysign** : crée une empreinte digital (signature) utilisée lors de l'authentification par relation de confiance
      - **HostbaseAuthentication**: à activer aussi du côté client

```
Host *
        ForwardX11 yes
        EnableSSHKeysign yes
        HostbasedAuthentication yes
```

Champs nécessaires du côté client

Côté serveur : changement de clés

- Changement de clés du serveur
  - Commande : ssh-keygen

    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 reconnexion alors que la clé du serveur a changé :

```
[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 : changement de clés

- Conséquence d'un changement de clés du serveur
  - 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 : généralités

- Sshd en écoute sur le port 22/TCP

```
[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

Authentification forte (AF)

- SSH et l’authentification forte …

Authentification forte

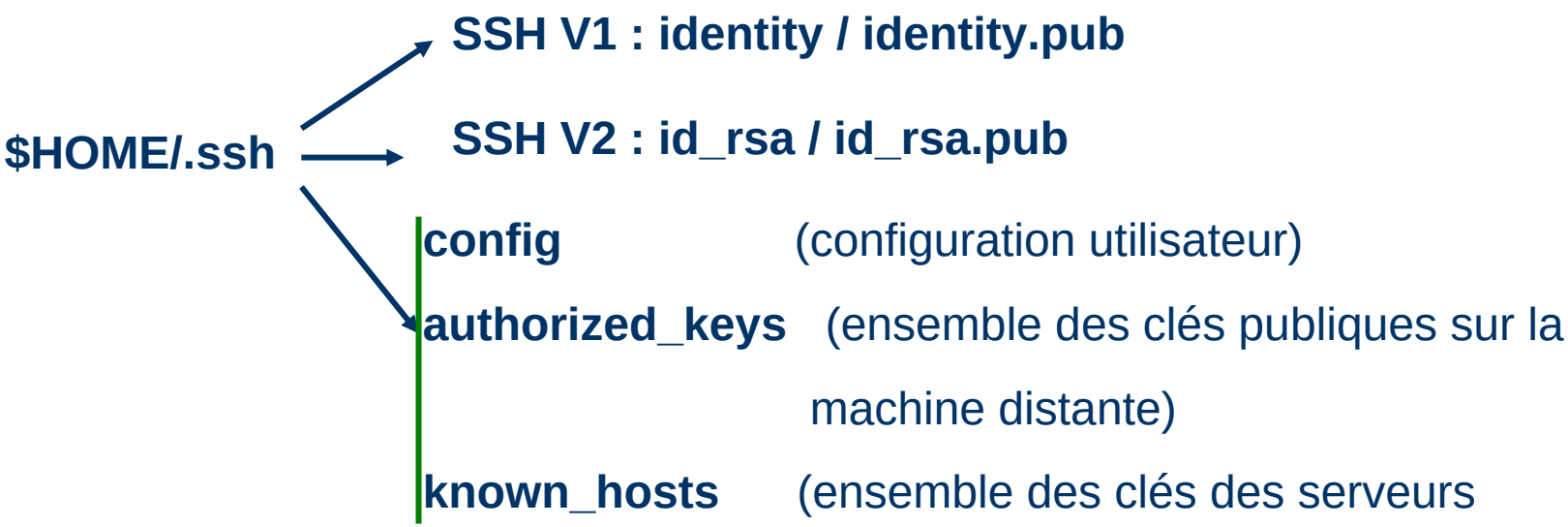
- Connexion par authentification forte
  - Basée sur une procédure d'identification plus complexe et différente de celle des systèmes Unix classiques *(nom et mot de passe)*
  - Cette procédure repose sur un principe d'une paire de clés publique/privée dont la clé privée est protégée par une phrase d'identification
    - **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
  - Structure des fichiers impliqués :



AF : gestion des clés - Unix

- Gestion des clés et agents
  - Répertoire $HOME/.ssh avec les droits des fichiers

```
[fbongat@localhost .ssh]$ ll
total 80
-rw-r--r--    1 fbongat  lmd-ens        0 mai 26  2003 authorized_keys
-rw-------    1 fbongat  lmd-ens     1876 déc 16 08:08 config
-r--------    1 fbongat  lmd-ens      530 jan 21  2003 identity
-rw-r--r--    1 fbongat  lmd-ens      334 oct 31  2002 identity.pub
-r--------    1 fbongat  lmd-ens      951 jan 21  2003 id_rsa
-r--------    1 fbongat  lmd-ens      951 mar  6  2003 id_rsa_ipslmd
-rw-r--r--    1 fbongat  lmd-ens      223 mar  6  2003 id_rsa_ipslmd.pub
-rw-r--r--    1 fbongat  lmd-ens      226 oct 24  2002 id_rsa.pub
-rw-r--r--    1 fbongat  lmd-ens    47617 mar  5 11:58 known_hosts
-rw-------    1 fbongat  lmd-ens      512 jui 15  2003 random_seed
[fbongat@localhost .ssh]$ _
```

AF : gestion des clés - Unix

- Gestion des clés et agents
  - 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
  - Création des paires de clés (SSH V1)

```
[root@localhost root]# ssh-keygen -t rsa1
Generating public/private rsa1 key pair.
Enter file in which to save the key (/root/.ssh/identity):
Enter passphrase (empty for no passphrase): *******
Enter same passphrase again: *******
Your identification has been saved in /root/.ssh/identity.
Your public key has been saved in /root/.ssh/identity.pub.
The key fingerprint is:
0f:76:d1:ea:b6:6b:12:8e:45:69:45:3d:7b:c0:80:77 root@localhost
[root@localhost root]#
[root@localhost root]# ll .ssh
total 8
-rw-------    1 root     root          529 mar 11 23:22 identity
-rw-r--r--    1 root     root          333 mar 11 23:22 identity.pub
[root@localhost root]#
[root@localhost root]#
```

Création d'une paire de clé

phrase d'identification

Nom des fichiers de clés

AF : gestion des clés - Unix

- Gestion des clés et agents
  - Création des paires de clés (SSH V2)

```
[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

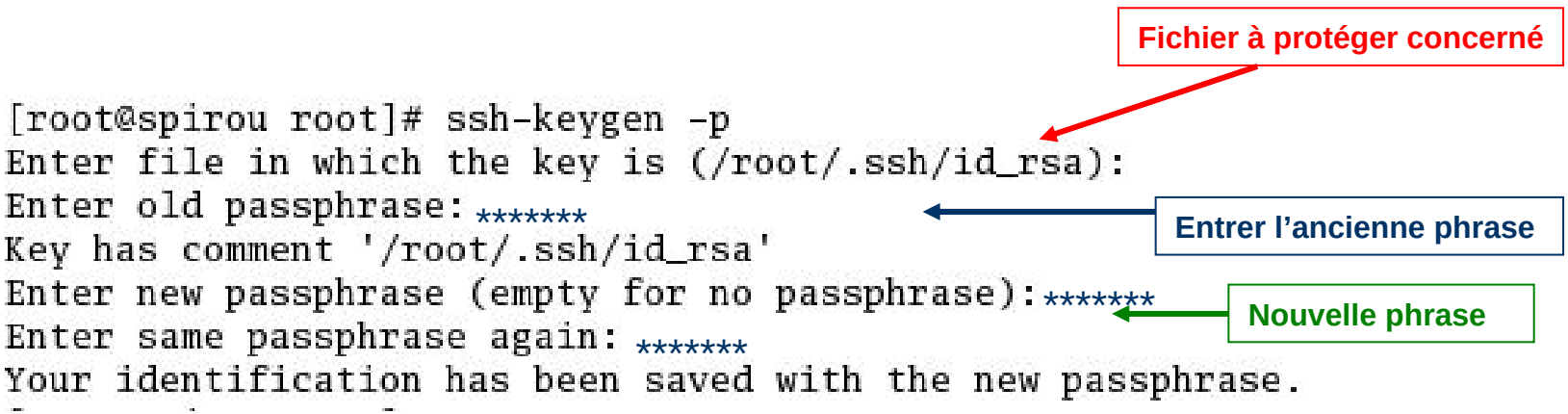
  - On peut de la même manière générer des clés avec un autre algorithme de cryptage : DSA

    ssh-keygen –t dsa

AF : gestion des clés - Unix

- Gestion des clés et agents
  - Changement de phrase d'authentification

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
      - On dispose 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
      - Les droits de ce fichier sont uniquement à l'utilisateur
    - Soit envoyer le fichier (clé publique) à l'administrateur du serveur distant qui l'installera sur la machine

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 authorized_keys :
    - Copier le fichier (clé publique) via un transfert de fichier *ssh classique* (scp ou sftp) si la connexion est autorisée

```
[roo spirou .ssh]# scp id_rsa.pub fbongat@spip:.ssh/
fbongat@spip's password:
*******
id_rsa.pub                                    100%  221    853.0KB/s   00:00
[root@spirou .ssh]#
[root@spirou .ssh]# ssh fbongat@spip
fbongat@spip's password:
[fbongat@spip fbongat]$ cd .ssh
[fbongat@spip .ssh]$ ls
id_rsa.pub  known_hosts
[fbongat@spip .ssh]$
[fbongat@spip .ssh]$ cat id_rsa.pub >> authorized_keys
[fbongat@spip .ssh]$ chmod 600 authorized_keys
[fbongat@spip .ssh]$ ls
authorized_keys  id_rsa.pub  known_hosts
```

Envoi de la clé publique

Connexion sur la machine distante

Insertion de la clé publique dans l'album des clés publiques

AF : gestion des clés - distribution

- Mise en place des clés
  - 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 – import/export

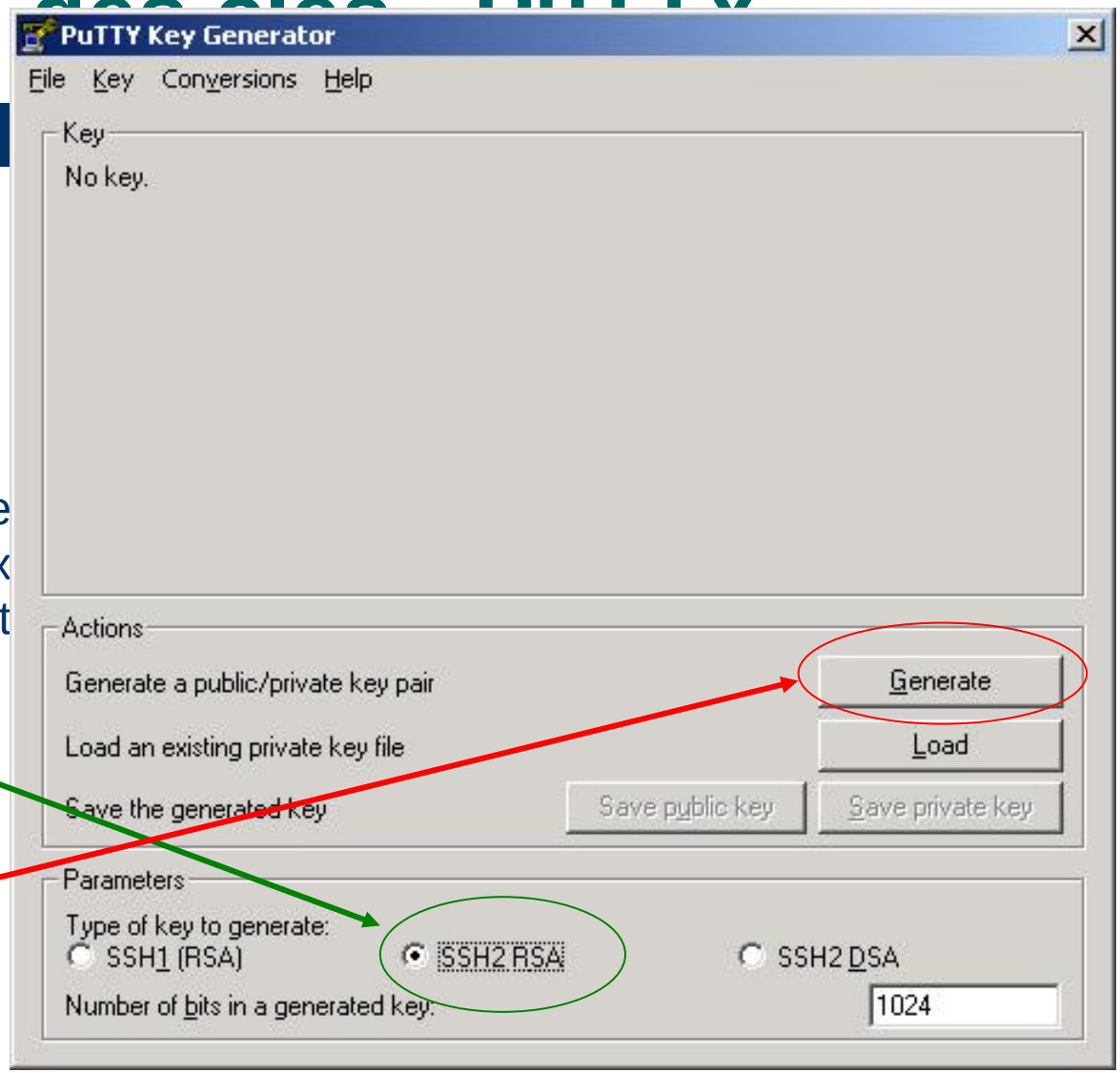
- Import/Export de clé Windows/Unix (ssh-keygen –i/-e)
  - méthode d'authentification par passphrase (avec un couple de clés publique/privée) à partir d'un PC Windows utilisant PuTTY ou SSH client, voici la meilleure manière de procéder.
    - A partir du poste Windows générer une nouvelle paire de clés publique/privée (appelées par exemple id_cle_window et id_cle_window.pub)
    - transférez la clé publique (id_cle_window.pub) sur la machine Unix
    - connectez vous sur la machine cible, et effectuez (ou demandez à l'administrateur d'effectuer les commandes suivantes) :
      - # **ssh-keygen -i -f** id_cle_window.pub > id_cle_unix.pub
      - # cat id_cle_unix.pub >> authorized_keys
  - Et maintenant, vous voici prêt à vous connecter par authentification forte à partir des clients Windows puTTY et SSH (ssh.com)
  - Dans l'autre sens l'export de clés unix : **ssh-keygen -e**

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

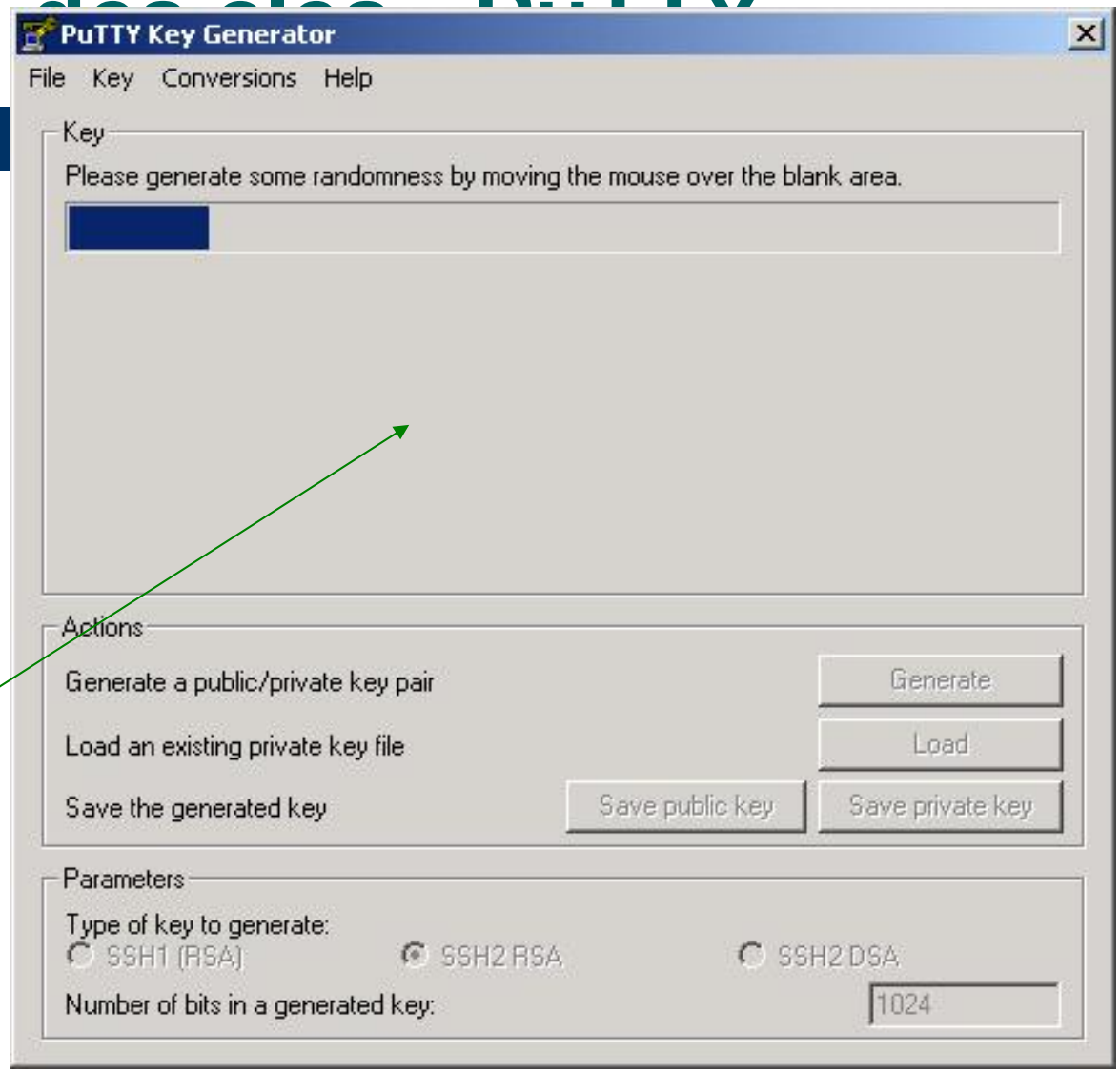
- Gestion des clés et agents: PuTTY
  - Création des paires de clés avec PuTTY
    - Sélectionner le type de clés à créer (sous Unix le plus souvent ce sont les clés v2 RSA qui sont utilisées)
    - Lancer la génération du couple de clés (publique/privée)



AF : gestion des clés - PuTTY

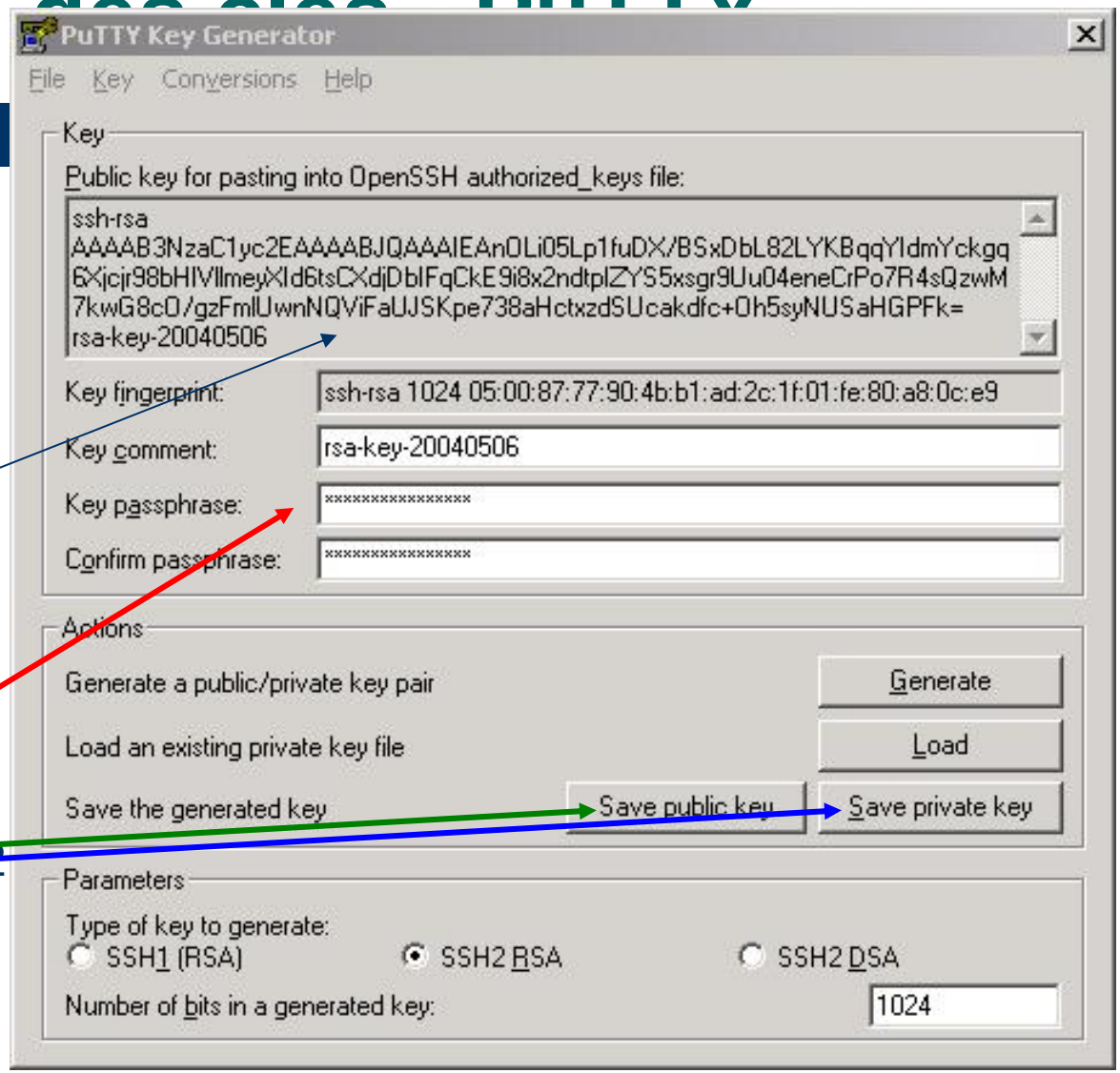
- Gestion des clés et agents: PuTTY
  - Création des paires de clés avec PuTTY
    - Pour générer les clés, il faut bouger la souris dans le cadre sous la barre de progression

Le mouvement de la souris dans ce cadre permet de générer de l'aléatoire pour les algorithmes créant les clés



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

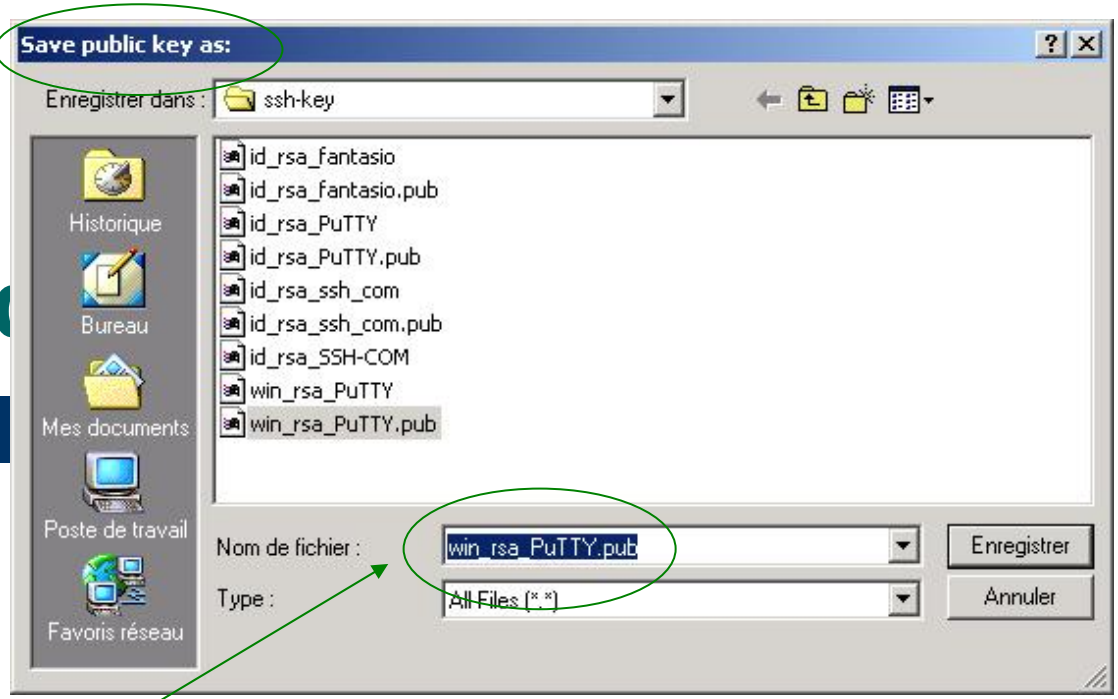


AF : gestion c

- 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

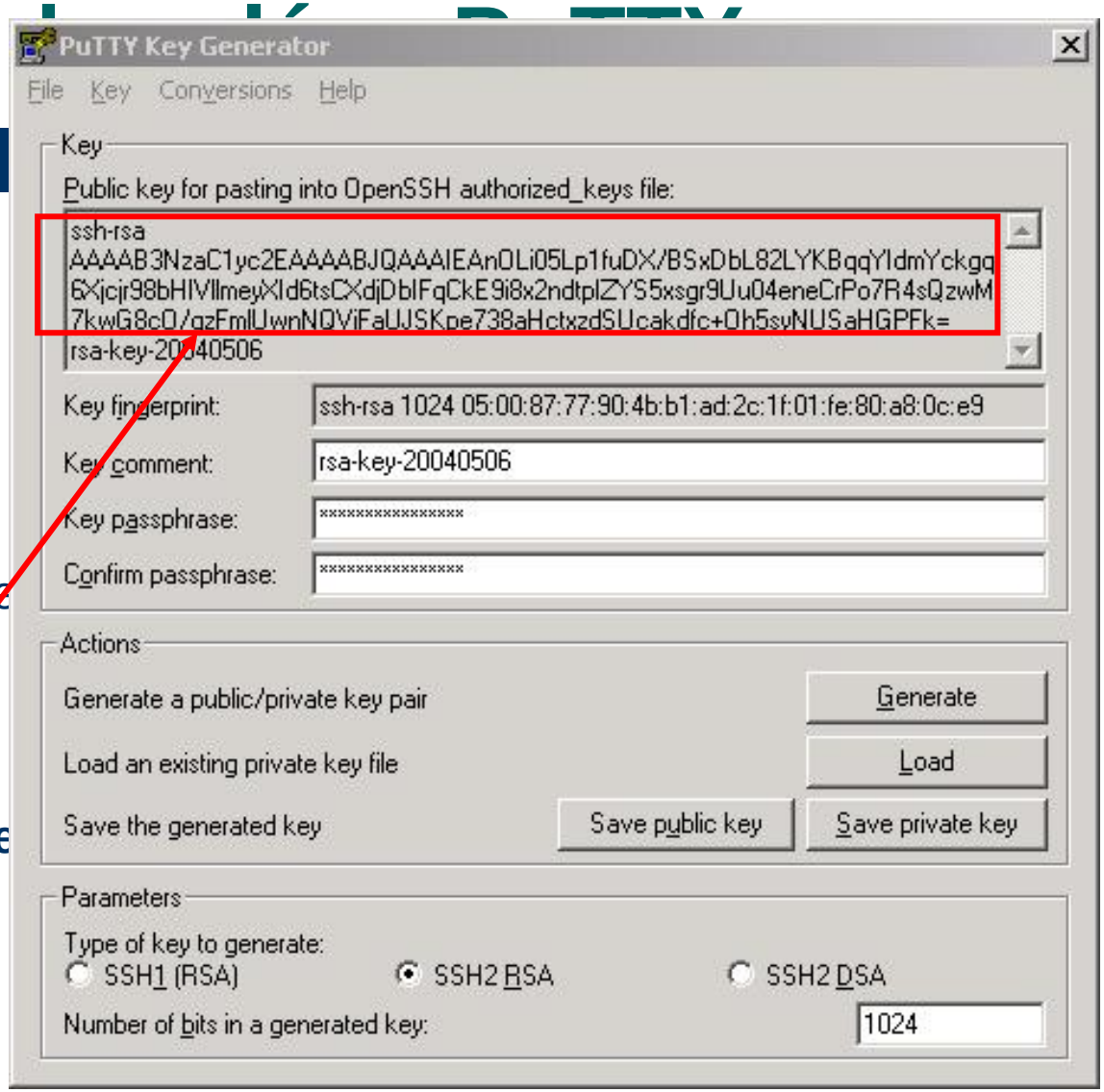
- Mise en place des clés PuTTY
  - La clé publique sur le serveur distant :
    - ***Copier/coller*** du cadre rouge sur **1 ligne** dans le fichier ***authorized_keys*** la machine distante

    (ssh-rsa AAAB3…..k=)

    Vérification de cohérence

    Sur linux :

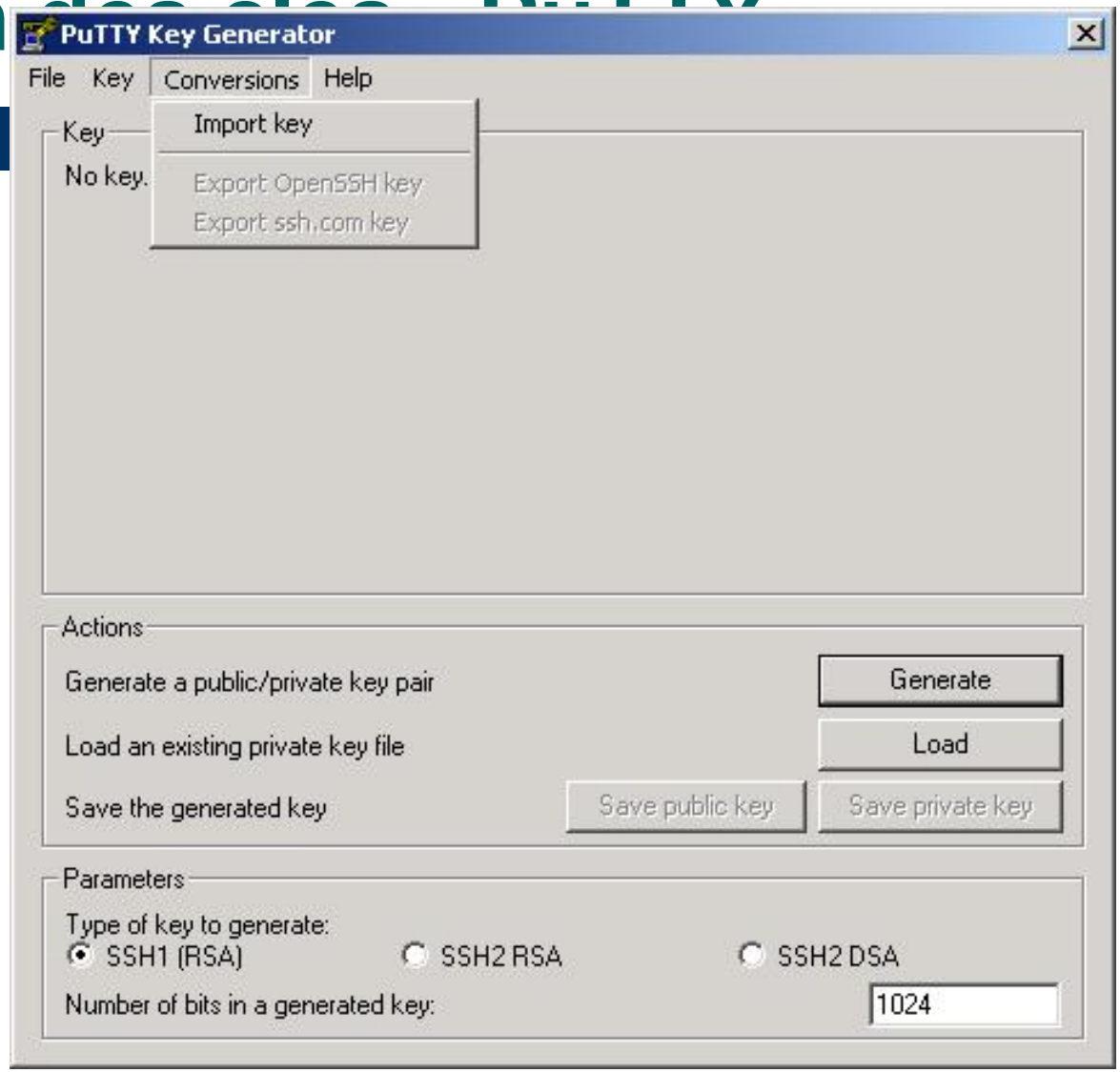
    ssh –l –f ~/.ssh/authorized_keys



AF : gestion des clés - PuTTY

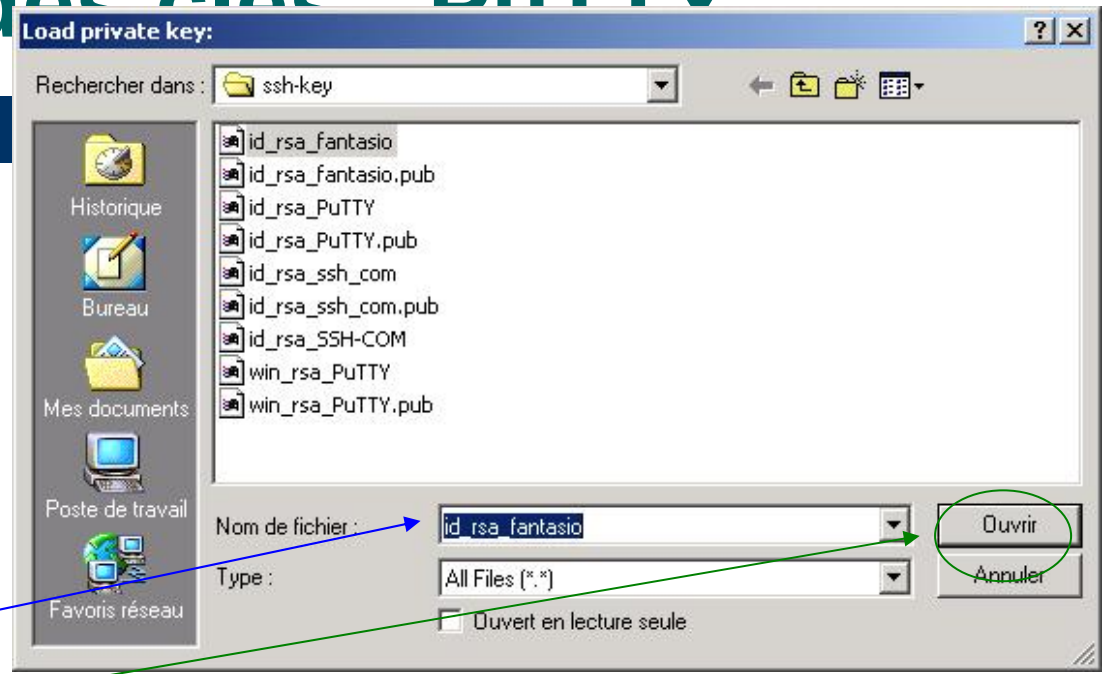
- Gestion des clés : PuTTY
  - Importation de clés provenant d'environnement Unix

  - Lancer le programme ***puttygen***
  - Aller dans le Menu « ***conversions*** »
  - Puis « ***Import Key*** »



AF : gestion des clés - PuTTY

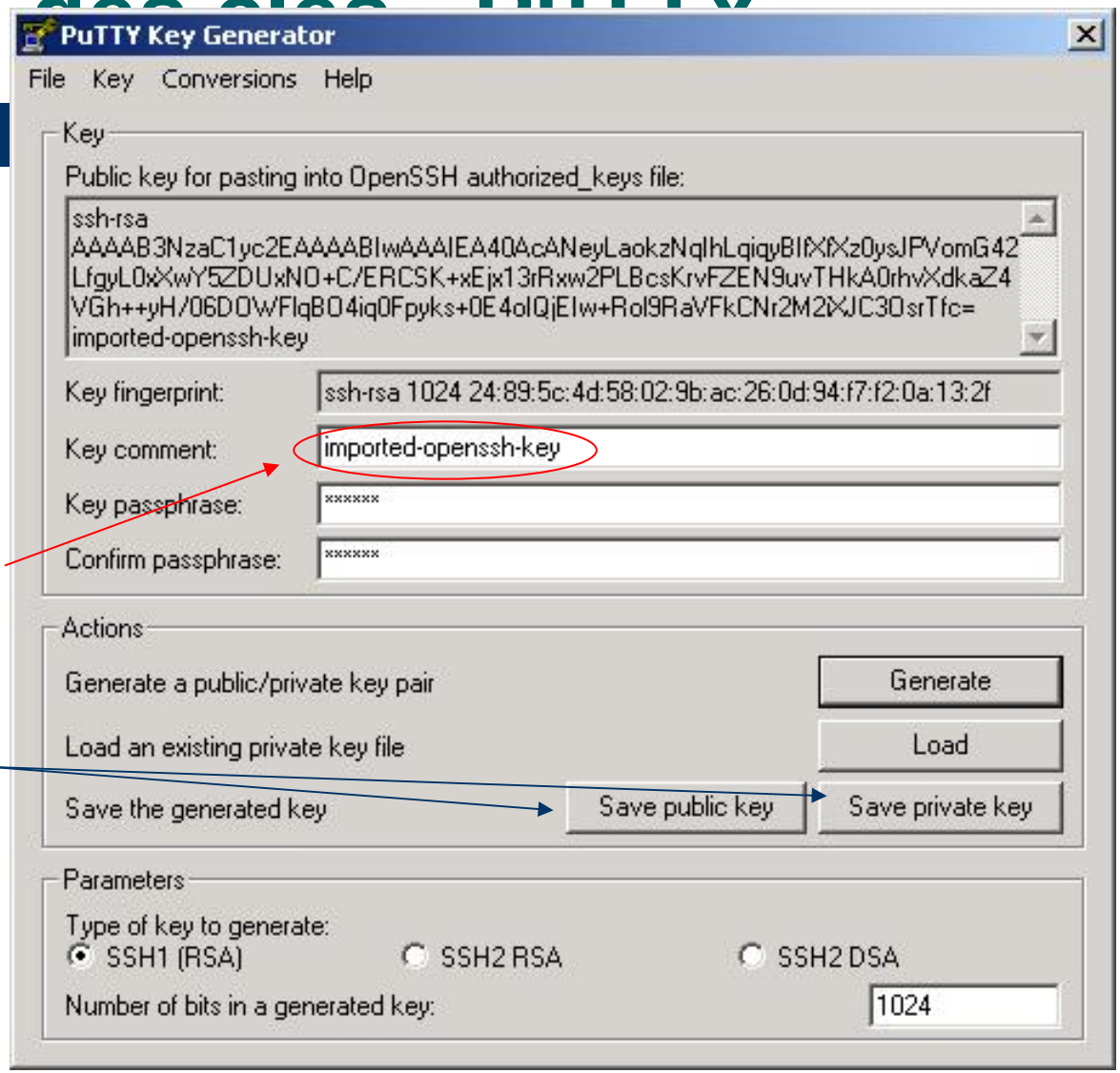
- Gestion des clés : PuTTY
  - Importation de clés provenant d'environnement Unix
    - Chargement de la clé privée importée
    - Valider par « ***ouvrir*** »
    - Entrer la phrase d'authentification protégeant la clé



AF : gestion des clés - PuTTY

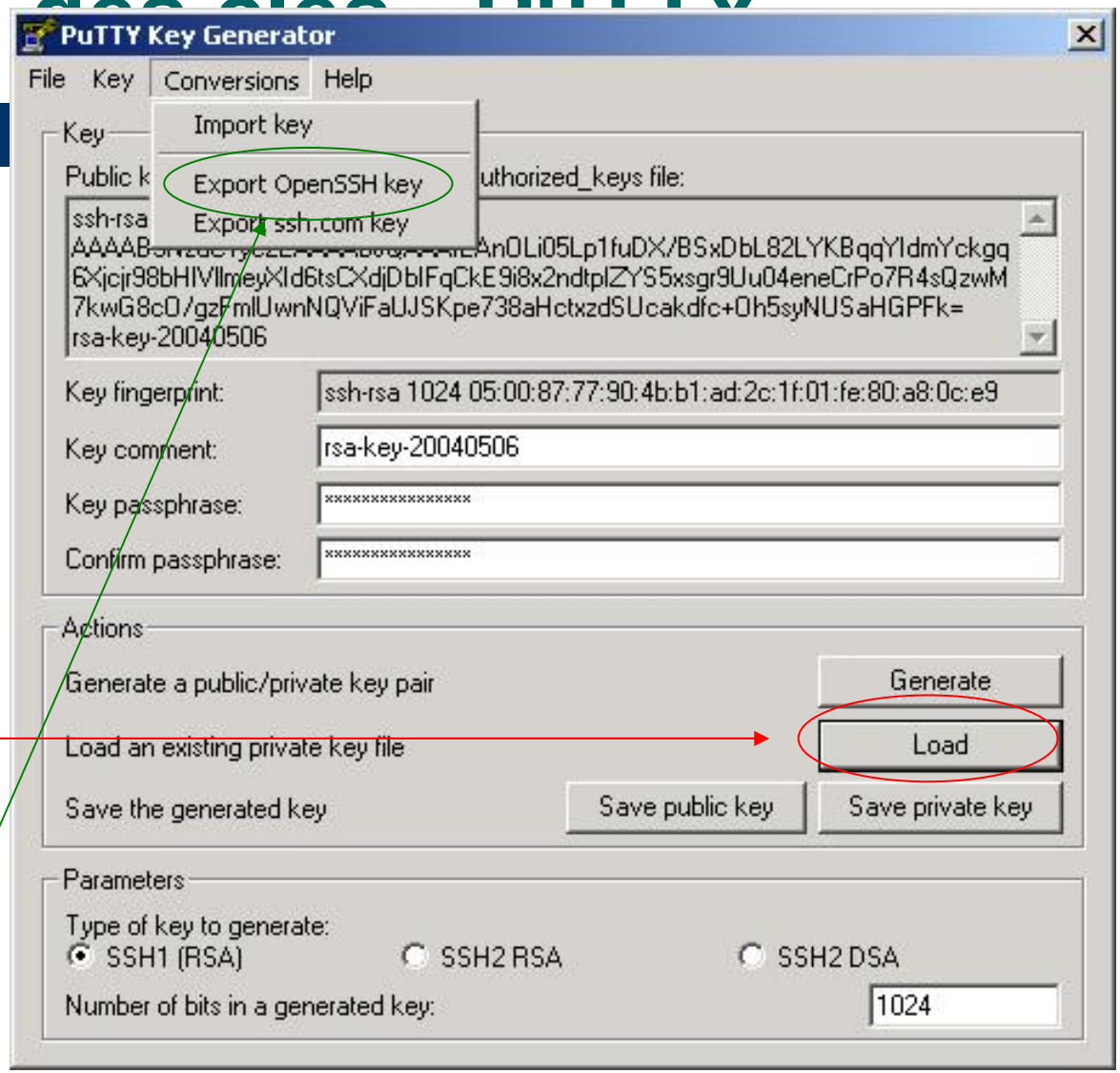
- Gestion des clés : PuTTY
  - Importation de clés provenant d'environnement Unix
    - Vérification de l'import
    - Il ne reste plus qu'à sauver les nouvelles clés au format PuTTY

      (voir création des clés avec PuTTY)



AF : gestion des clés - PuTTY

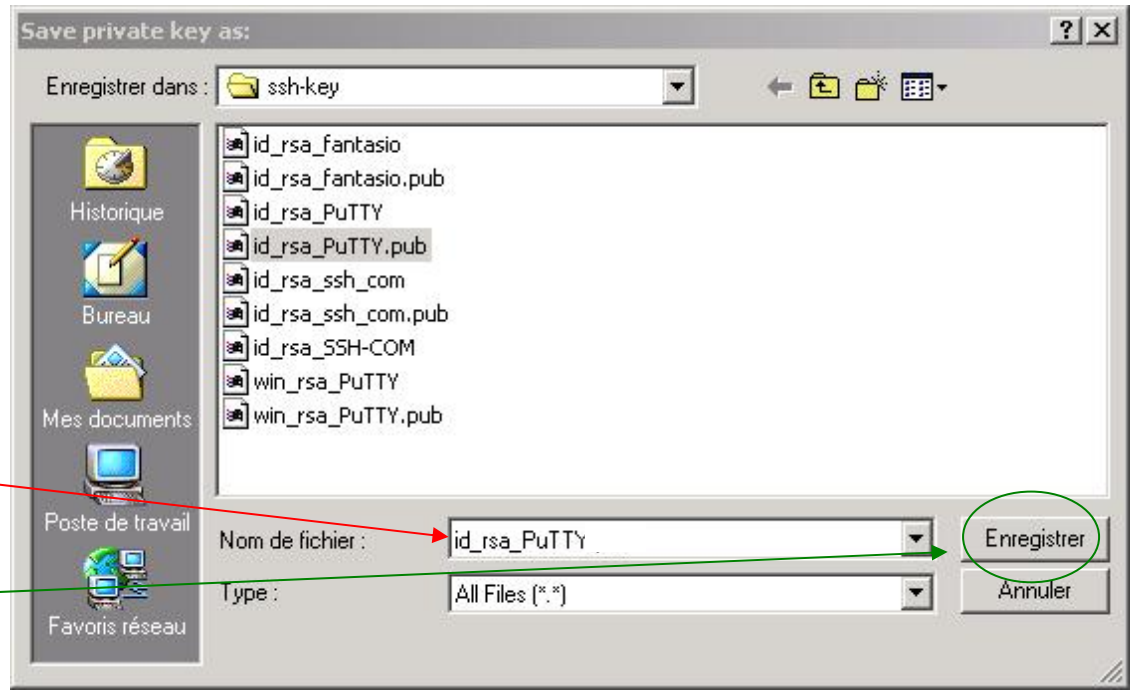
- Gestion des clés : PuTTY
  - Création et exportation d'une clé d'export PuTTY pour l'environnement Unix
    - Lancer ***puttygen***
    - Charger la **clé privée PuTTY** via le bouton « ***load*** »
    - Puis le menu « ***Conversions*** »
    - Sélectionner « ***Export OpenSSH key*** »



AF : gestion des clés - PuTTY

- Gestion des clés : PuTTY
  - Création et exportation d'une clé d'export PuTTY pour l'environnement Unix
    - Sauver la clé d'export dans un fichier en l'appelant afin de l'identifier

      (ici id_rsa_PUTTY)
    - Puis ***enregistrer***
    - Transférer la clé d'export sur une station Unix



AF : gestion des clés – PuTTY/Unix

- Gestion des clés: PuTTY
  - Création et exportation d'une clé d'export PuTTY pour l'environnement Unix
    - Se connecter sur le serveur Unix
    - Créer le couple de clés associées OpenSSH sur la station Unix à partir de la clé d'export avec :

      ssh-keygen –e puis ssh-keygen –i

      - Il faut d'abord convertir la clé d'export PuTTY en clé privée OpenSSH
      - Puis créer la clé publique à partir de la clé privée obtenue précédemment

AF : gestion des clés – PuTTY/Unix

- Gestion des clés et agents: PuTTY
  - Création et exportation d'une clé d'export PuTTY pour l'environnement Unix

Création de la clé privée OpenSSH grâce à la clé d'export

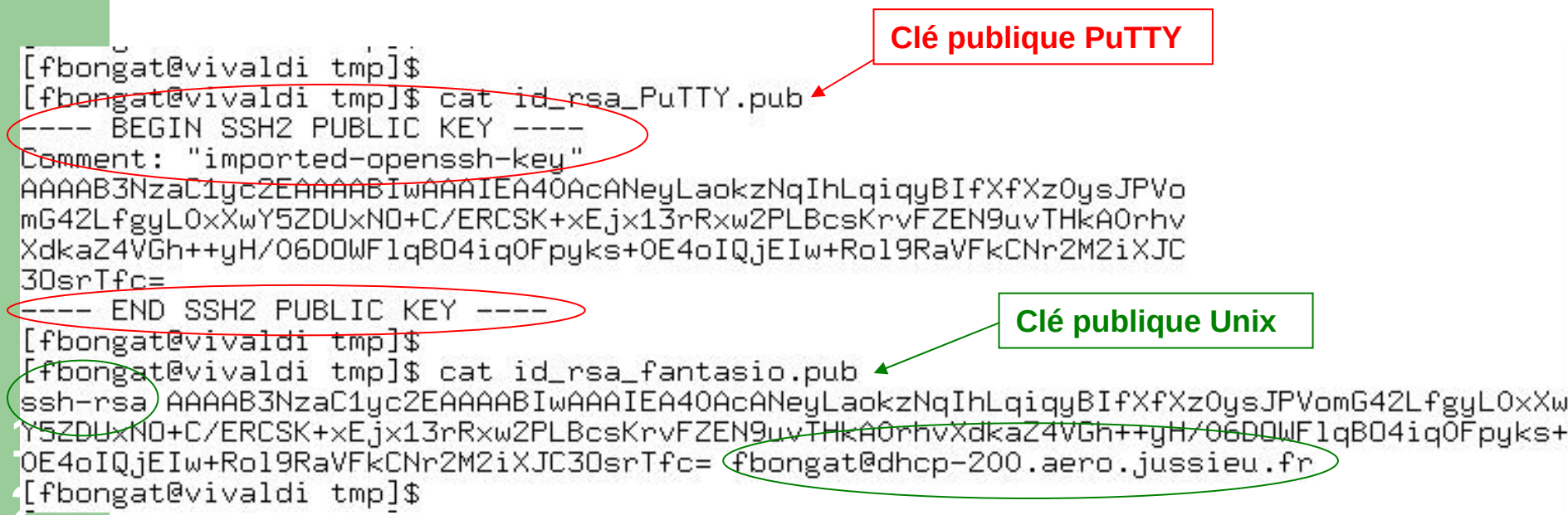
```
[fbongat@vivaldi tmp]$ ssh-keygen -e -f id_rsa_PuTTY > id_rsa_unix
Enter passphrase:
[fbongat@vivaldi tmp]$ ssh-keygen -i -f id_rsa_unix > id_rsa_unix.pub
[fbongat@vivaldi tmp]$
[fbongat@vivaldi tmp]$ cat id_rsa_unix.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAIEAnOLiO5Lp1fyDX/BSxDbL82LYKBqqYIdmYckgq6Xjcjr98bHIV
llmeyXId6tsCXdjDbIFqCkE9i8x2ndtpIZYS5xsgr9Uu04emeCrPo7R4sQzwM7kwG8cO/gzFmlUwnNQViFaUJ
SKpe738aHctxzdSUcakdfc+Oh5syNUSaHGPFk=
[fbongat@vivaldi tmp]$
[fbongat@vivaldi tmp]$ cat id_rsa_unix.pub >> ~/.ssh/authorized_keys
[fbongat@vivaldi tmp]$
```

Mise en place de la clé publique

Création de la clé publique OpenSSH à aprtir de la clé privée obtenue

AF : gestion des clés – PuTTY/Unix

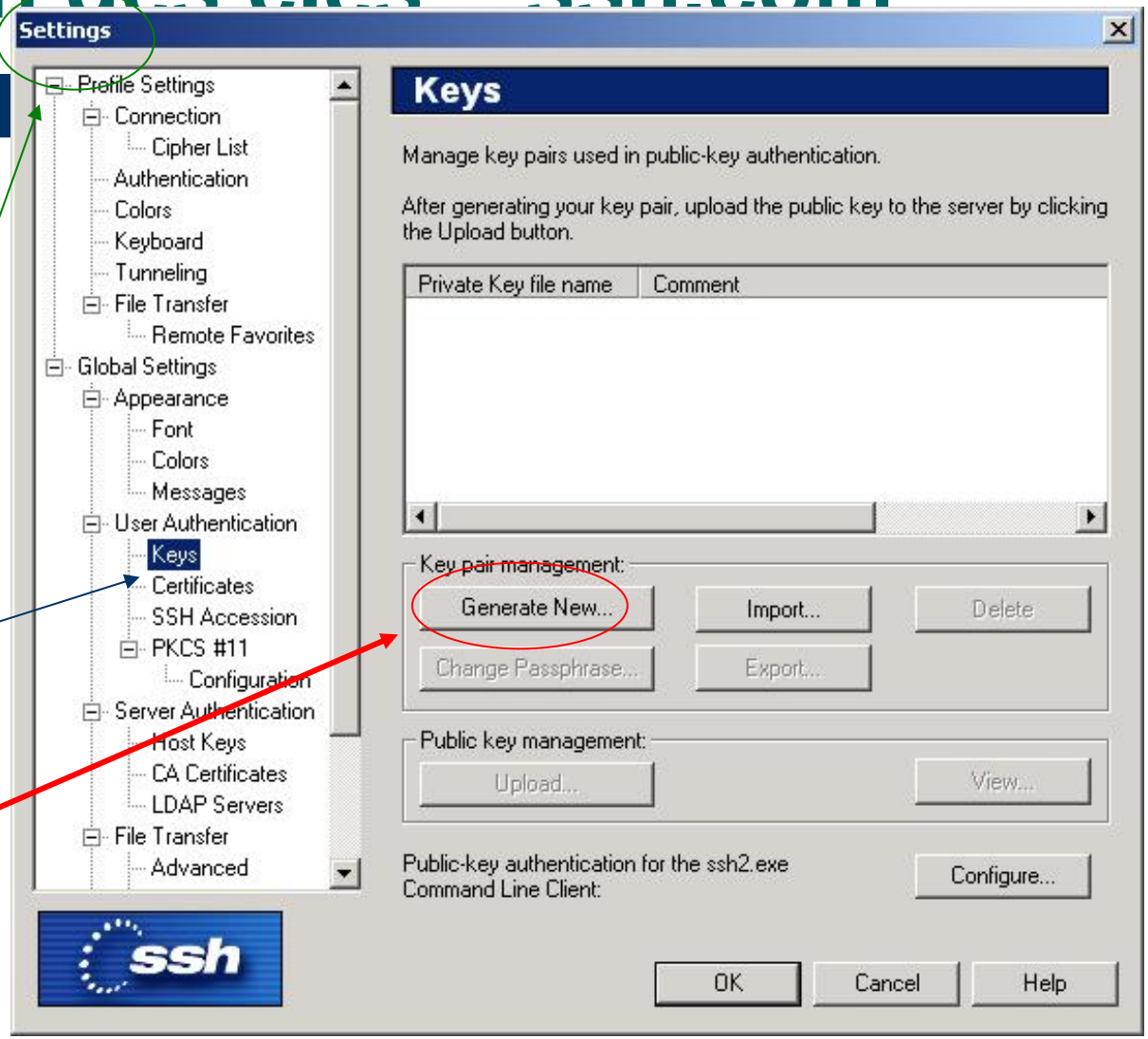
- Gestion des clés et agents: PuTTY
  - Différences du format des paires de clés Unix et PuTTY



AF : gestion des clés – ssh.com

- Gestion des clés avec Ssh.com Tectia

  - Lancer le programme ***ssh***
  - Menu ***Edit → Settings***
  - Sélectionner la valeurs : ***keys***
  - Lancer la création du couple de clés via ***Generate New***



AF : gestion des clés – ssh.com

- Gestion des clés avec Ssh.com Tectia
  - Sélection du type d'algorithme : RSA (par exemple)
  - La longueur de la clé

AF : gestion des clés – ssh.com

- Gestion des clés avec Ssh.com Tectia
  - Informations sur les clés :
    - Le nom de la clé privée (la clé publique aura automatiquement le même nom avec l'extension *.pub* en plus)
    - La phrase d'authentification protégeant l'accès à la clé privée

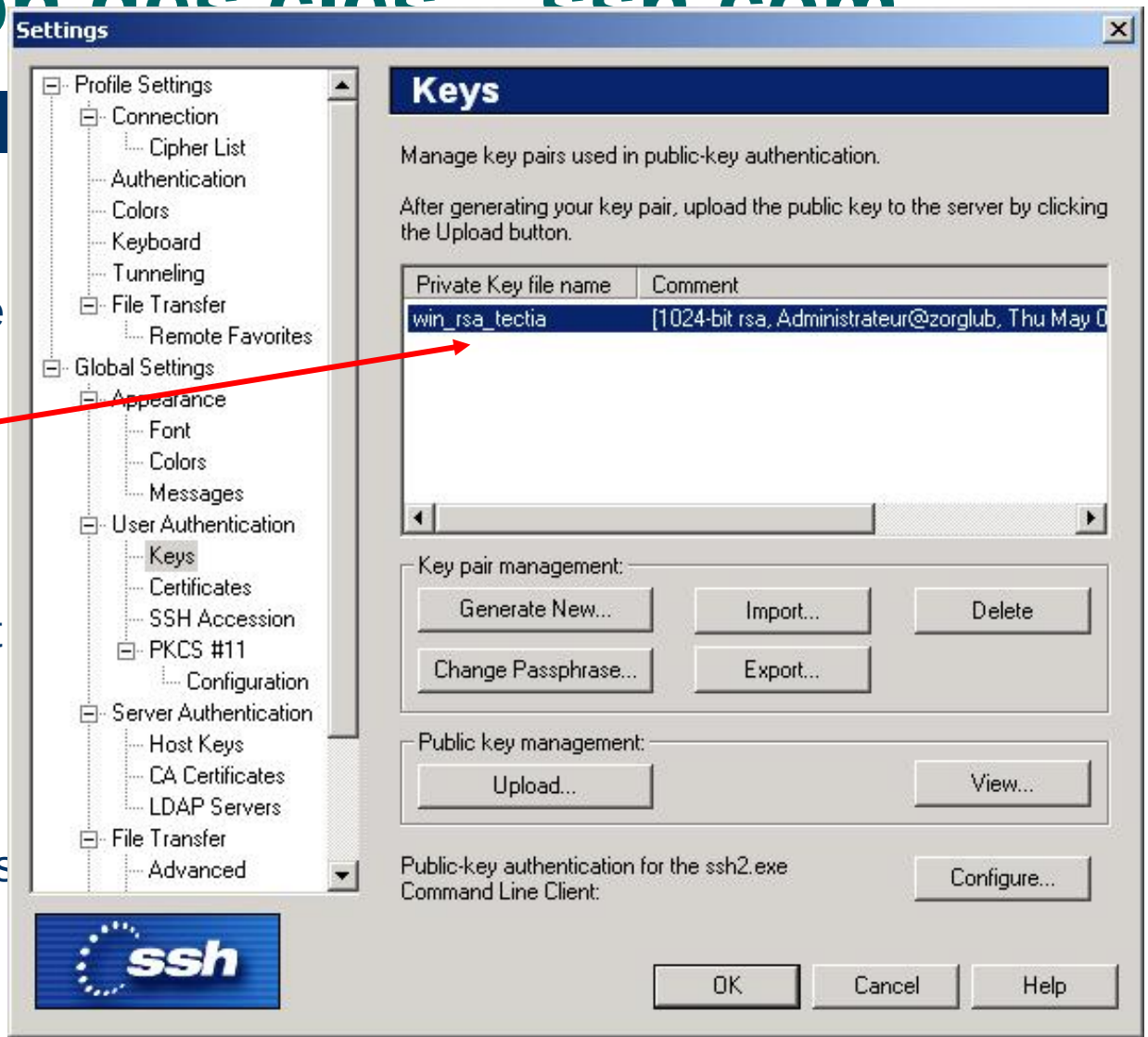
AF : gestion des clés – ssh.com

- Gestion des clés avec Ssh.com Tectia
  - Panneau rassemblant le nom et des informations sur les clés privées
  - Export de clé publique ssh.com vers OpenSSH uniquement :

Prendre la clé publique ssh.com, la transférer et utiliser la commande :

ssh-keygen –i –f win_ssh_tectia.pub > id_rsa_ssh_com.pub

Puis copier le contenu dans ***authorized_keys***



AF : connexion simple Unix

- Connexions par authentification forte
  - Pour forcer l'authentification forte d'un client vers un serveur lorsque celui-ci laisse tous les choix d'authentification, il suffit de configurer le fichier *config* en spécifiant la variable *RSAAuthentication* et de lui indiquer le nom du fichier clé privée :

```
Host fantasio
    User fbongat
    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]$
```

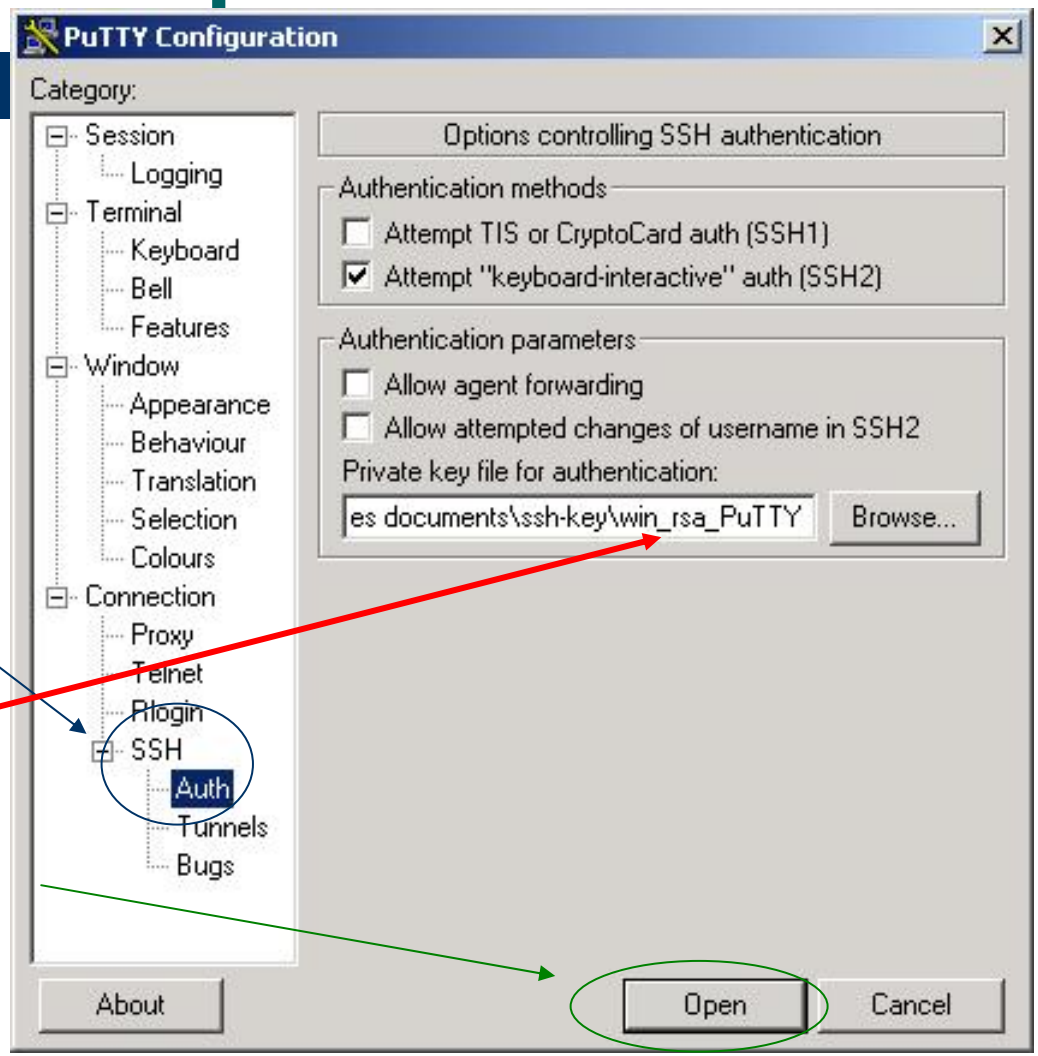
Phrase d'authentification

  - Par sftp :

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

AF : connexion simple PuTTY

- 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***



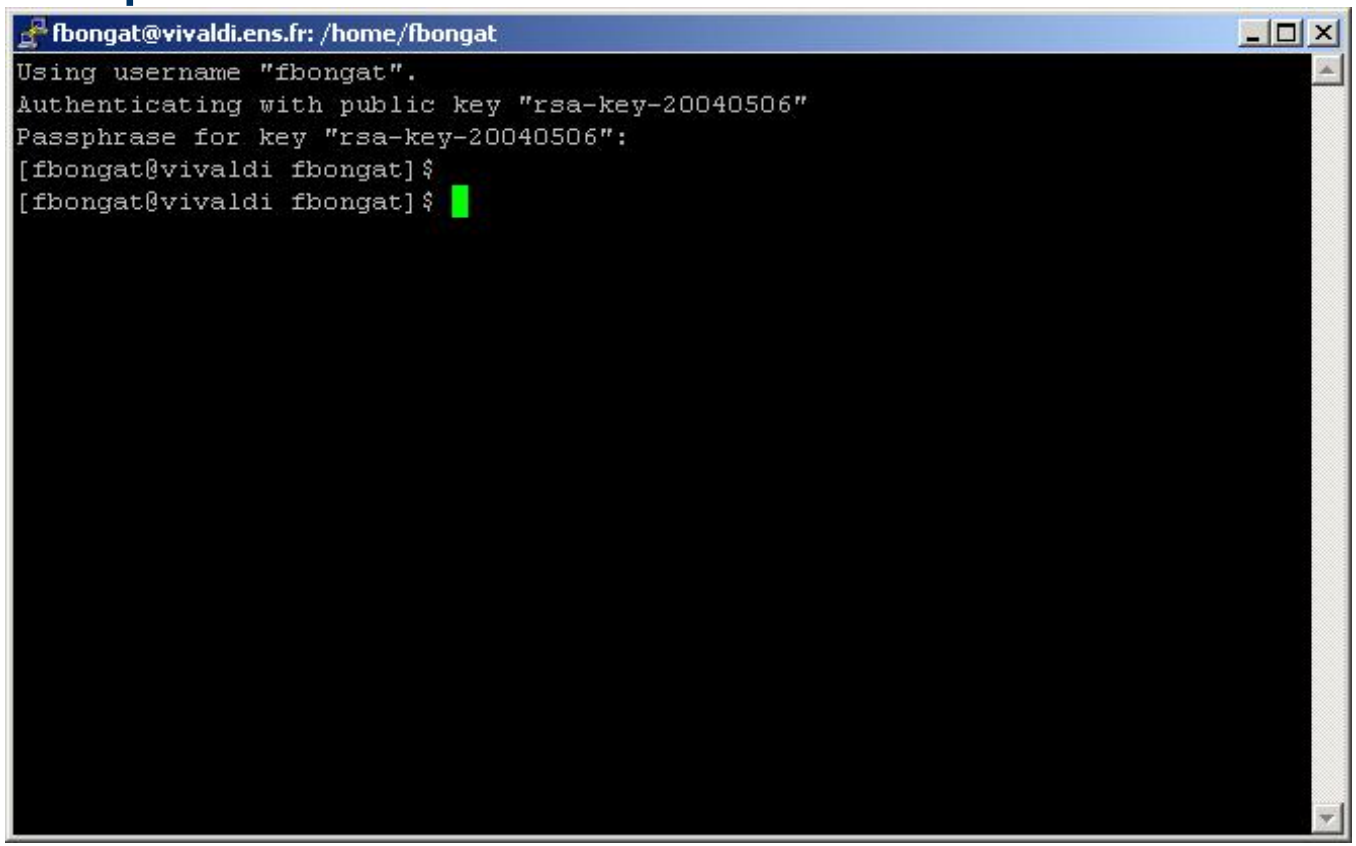
AF : connexion simple PuTTY

- Connexions par authentification forte

  par ssh

  PuTTY:

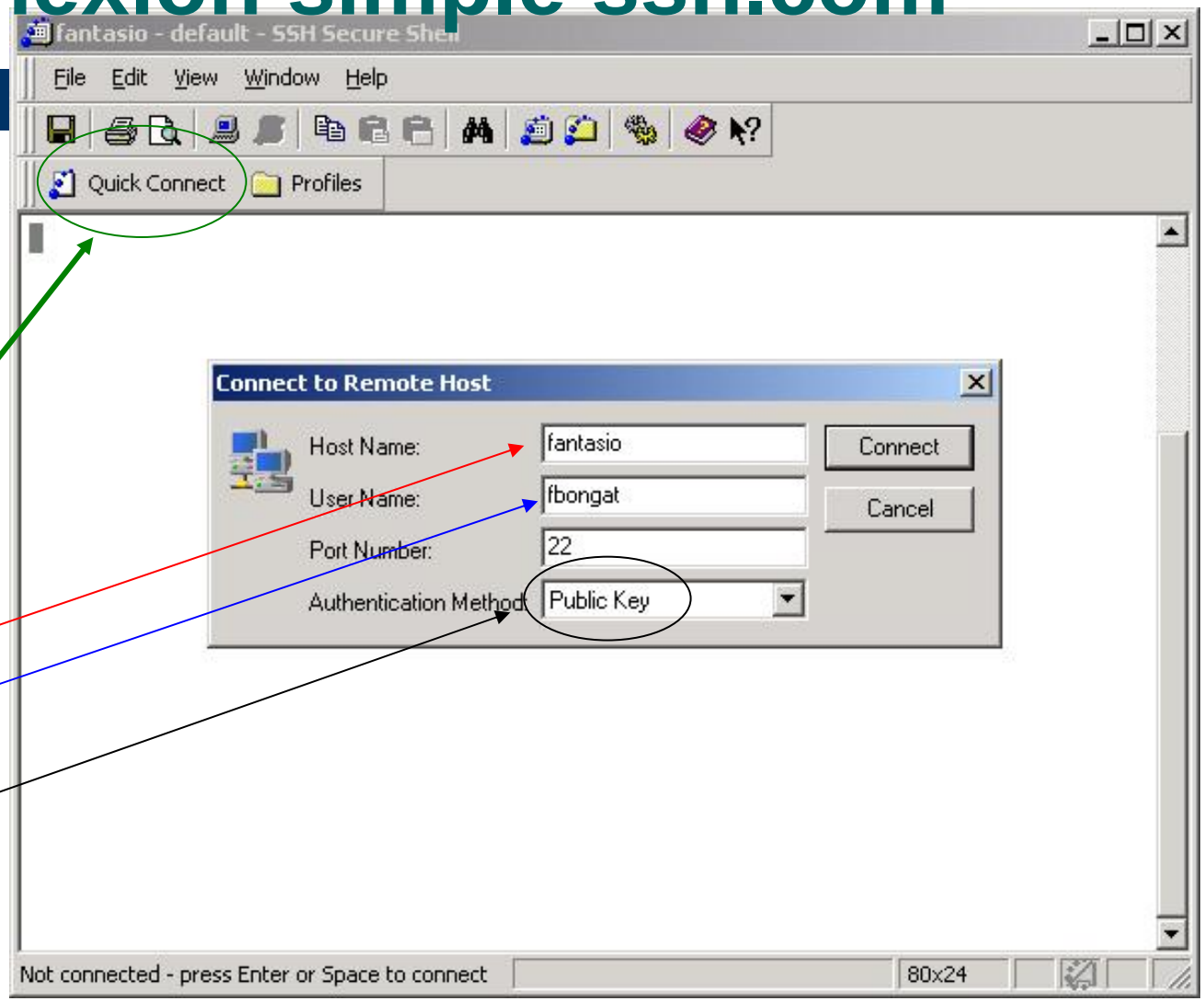
  Idem avec ***psftp*** et ***pscp***



AF : connexion simple ssh.com

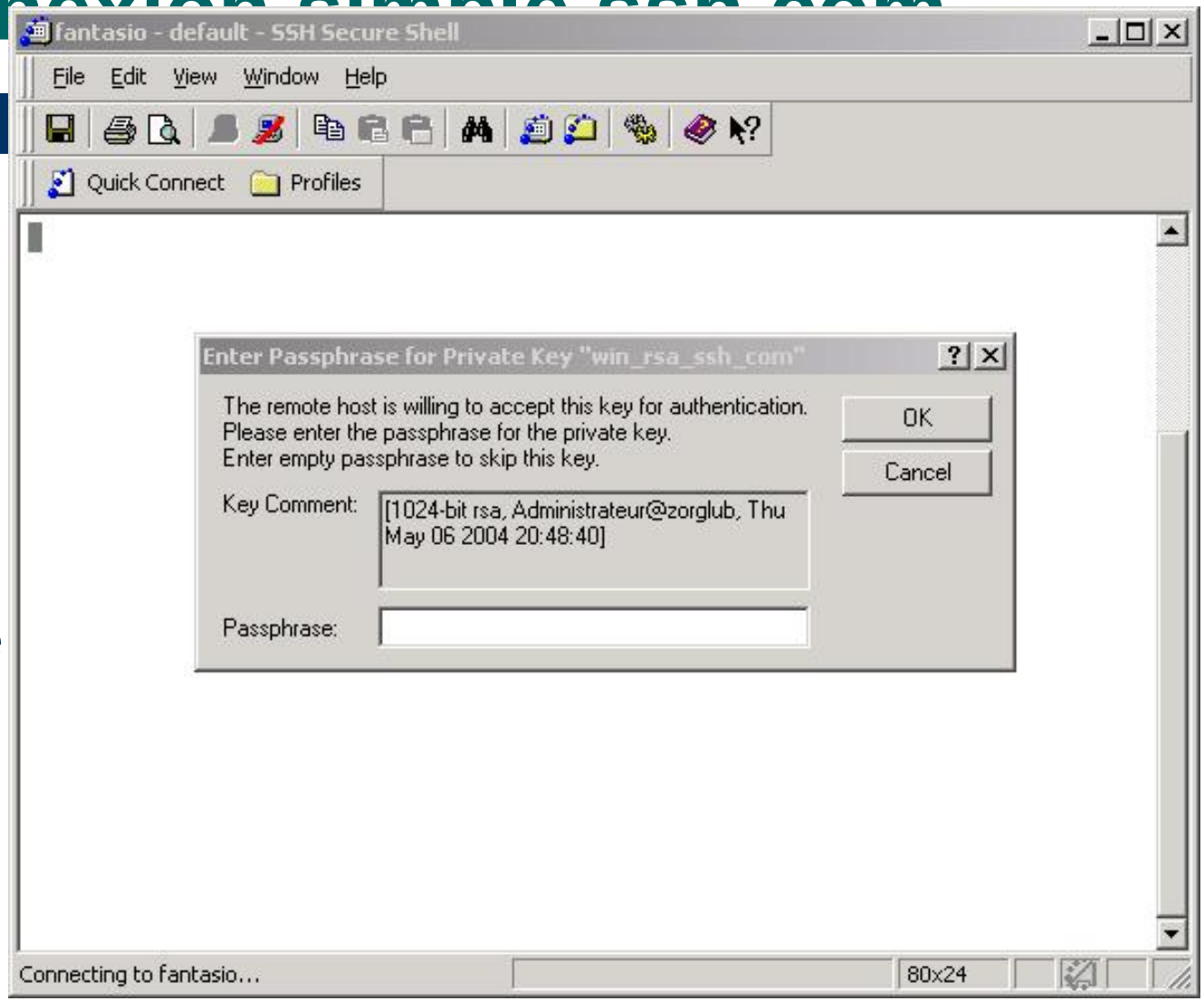
- SSH et Windows : ssh.com (Ssh Tectia)

  - Cliquer sur l'icône du client ssh
  - Puis sur :
  - ***Quick Connect***
  - Remplir la fenêtre qui apparaît
    - Nom de l'hôte distant
    - Login
    - Type de méthode d'accès : clé publique



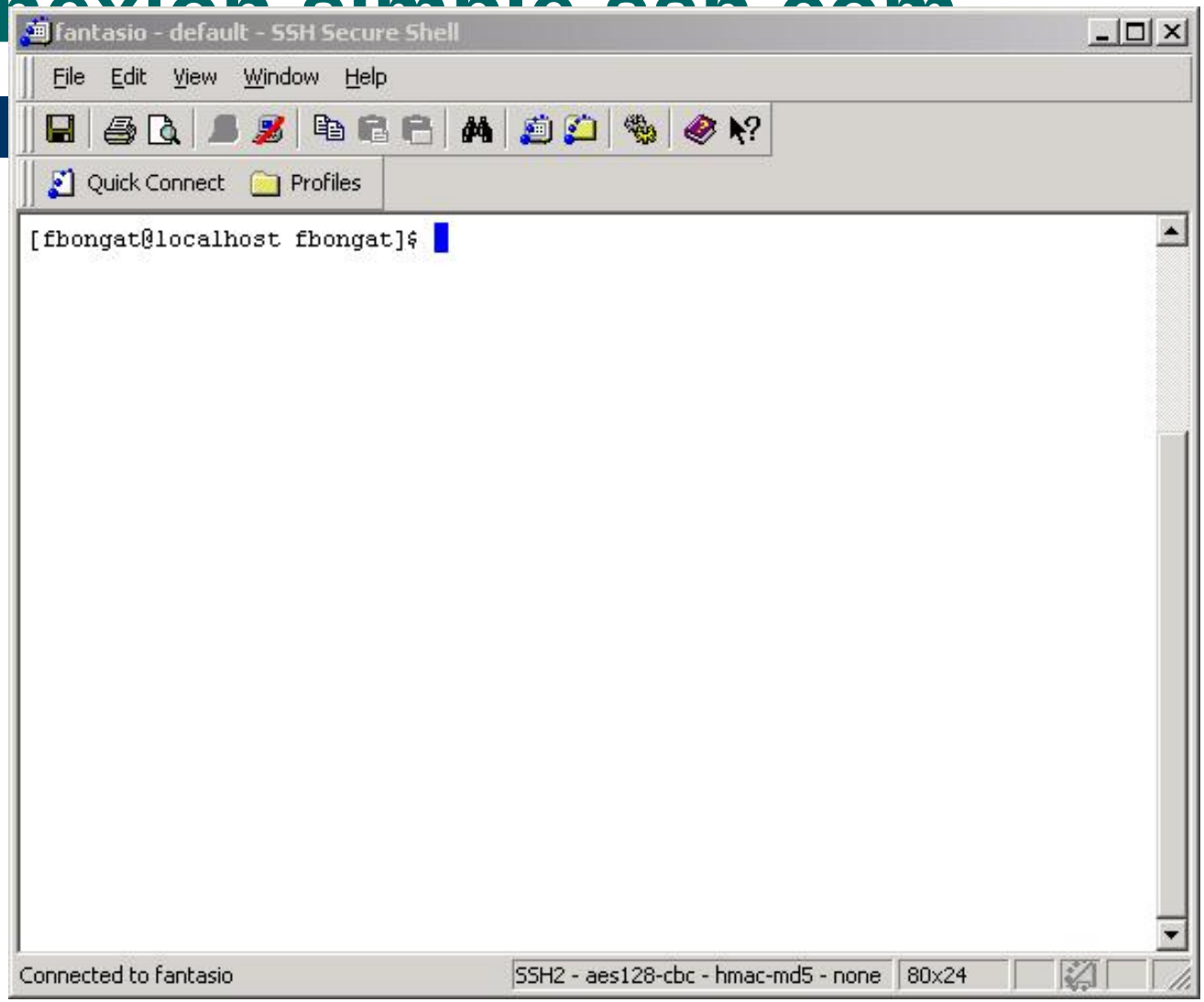
AF : connexion simple ssh.com

- SSH et Windows : ssh.com (Ssh Tectia)
  - Attendre la demande de la phrase d'authentification au lieu du mot de passe dans une boîte de dialogue qui va apparaître.



AF : connexion simple ssh.com

- SSH et Windows : ssh.com (Ssh Tectia)
  - Fenêtre de connextion ssh par authentification forte
  - Cela fonctionne aussi pour ***sftp***



AF : ssh agent

- Utilisation d'un agent
  - 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
  - Lancement d'un agent avec la commande ssh-agent dans un shell et stockage en mémoire avec la commande ssh-add

    # ssh-agent /bin/bash

    # 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 - Unix

- Gestion des clés et agents
  - **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 : ssh agent - KDE

- Gestion des clés et agents dans une session **KDE**
  - Le package *openssh-askpass-gnome.*.rpm* est nécessaire pour Gnome et Kde
  - 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
```

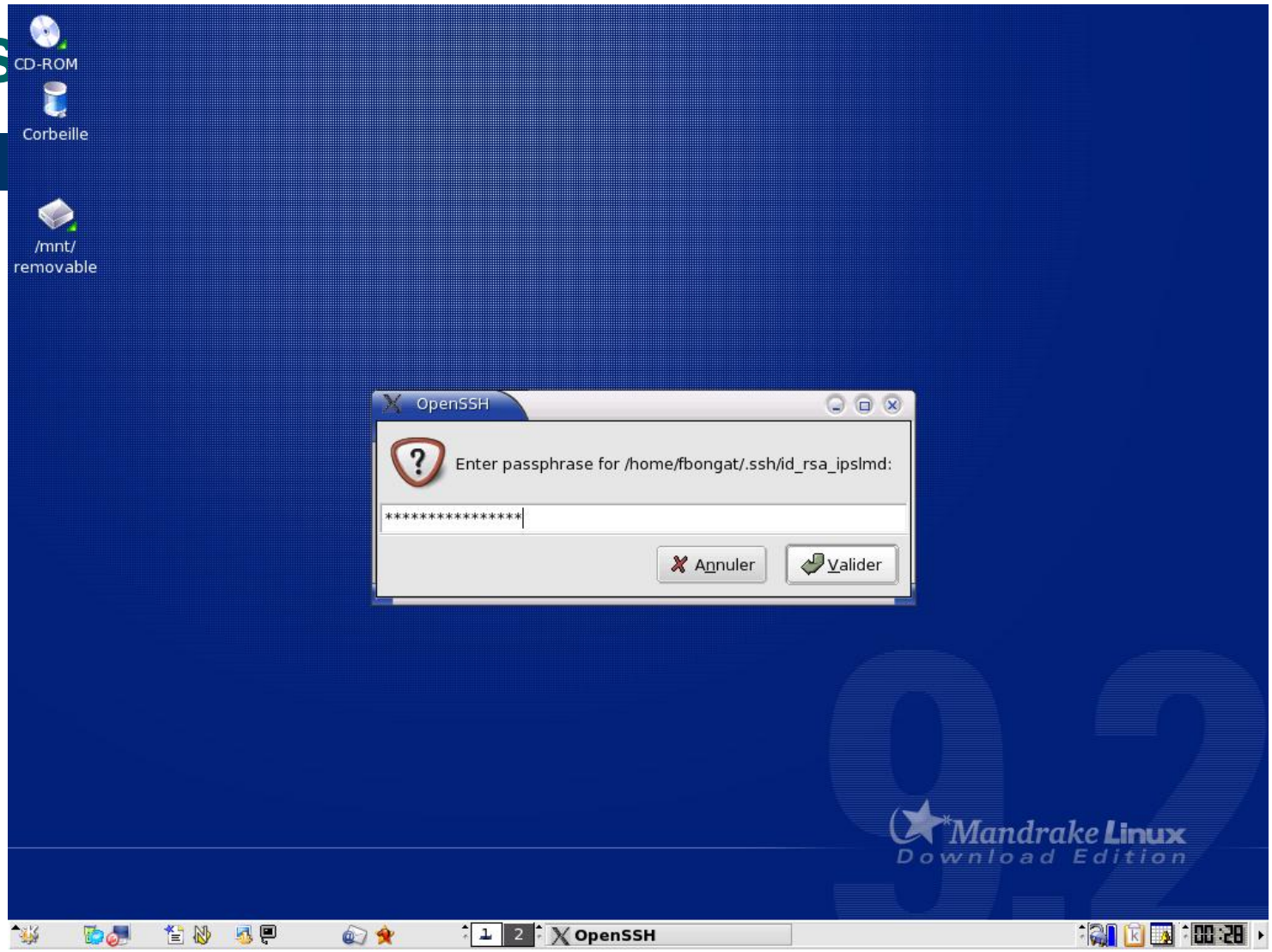
Script ssh-add

  - Une fois la session ouverte tous les connexions sont liées à l'agent.

AF : s

Gestion des clés et agents

Lancement dans l'environnement graphique (KDE)



AF : ssh agent - GNOME

- Gestion des clés et agents dans une session **Gnome**
  - Le package *openssh-askpass-gnome.*.rpm* est nécessaire pour Gnome et Kde
  - Lancement dans l'environnement graphique
    - ***ssh-agent*** lancé dans /etc/X11/Xsession (par défaut rien à faire)
    - Créer ou modifier fichier : ***$HOME/.gnome2/session-manual***

```
[root@spirou root]# cd .gnome2
[root@spirou .gnome2]# ls
accels  gdm  nautilus-scripts  panel2.d  session  session-manual  share
[root@spirou .gnome2]# cat session-manual

[Default]
num_clients=1
0,RestartStyleHint=3
0,Priority=60
0,RestartCommand=/usr/bin/ssh-add
[root@spirou .gnome2]# _
```

Informations à mettre dans le fichier

Fichier à modifier ou à créer

  - Une fois la session ouverte tous les connexions seront liées à l'agent.

SSH

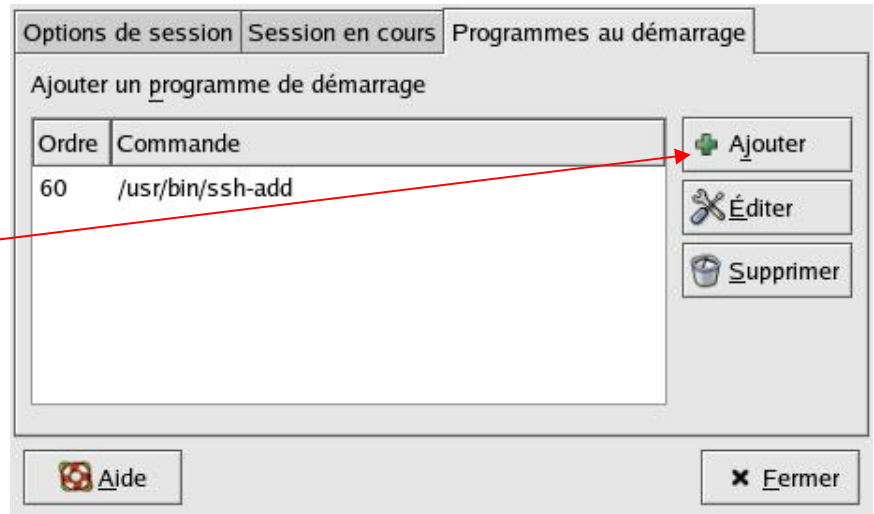
- Gestion des clés et agents dans une session **Gnome**
  - Autre méthode :
    - Via les menus : ***Préférences*** → ***Préférences Supplémentaires*** → ***Sessions*** → **puis l'onglet *Programme au démarrage***
    - Ou bien suivant les linux : ***Configuration*** → ***GNOME*** → ***Avancé*** → ***Sessions*** → ***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

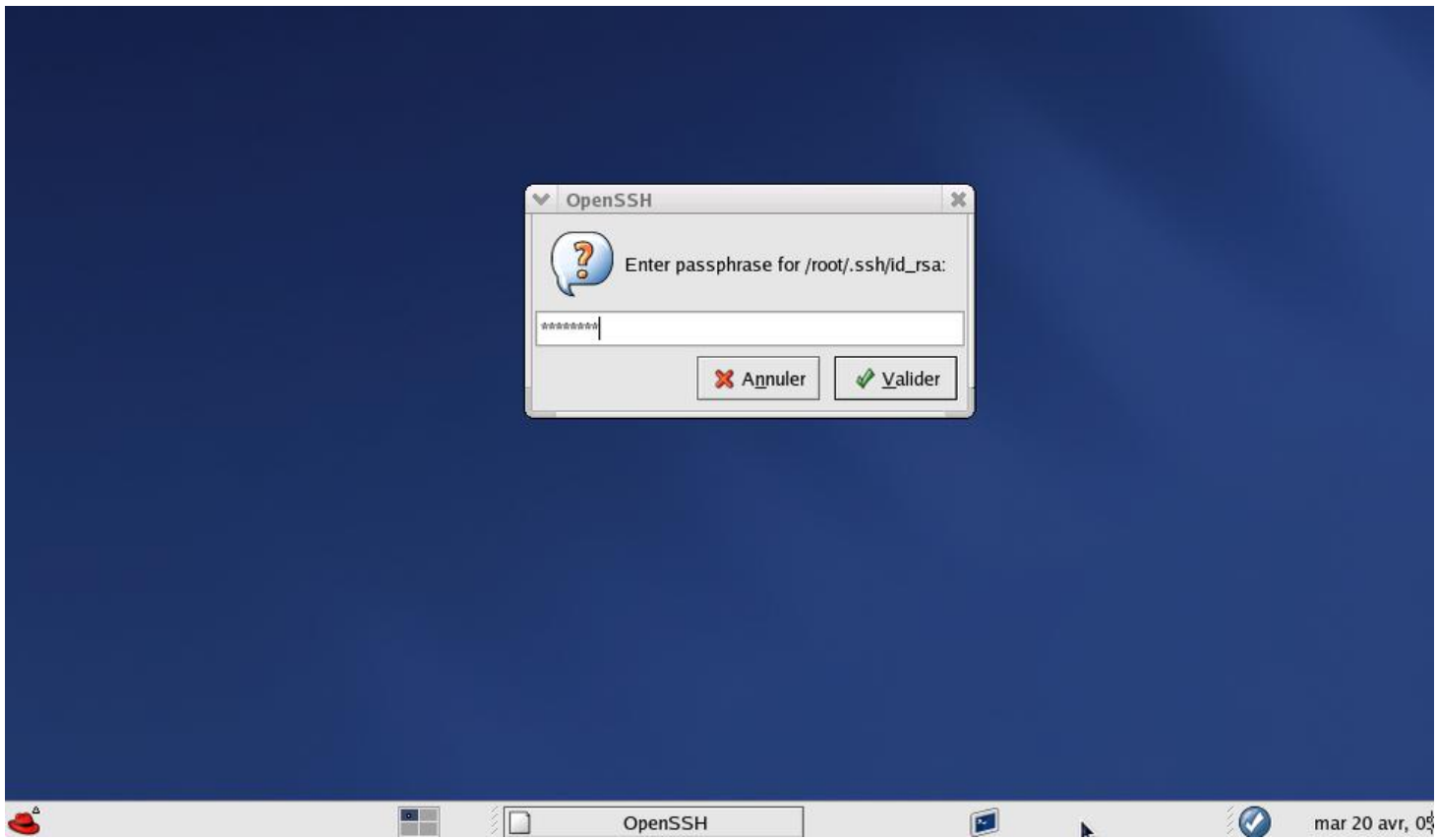


  - Une fois la session ouverte tous les connexions seront liées à l'agent.

AF : ssh agent - GNOME

Gestion des clés et agents

Lancement dans l'environnement graphique (Gnome)



AF : ssh agent - PuTTY

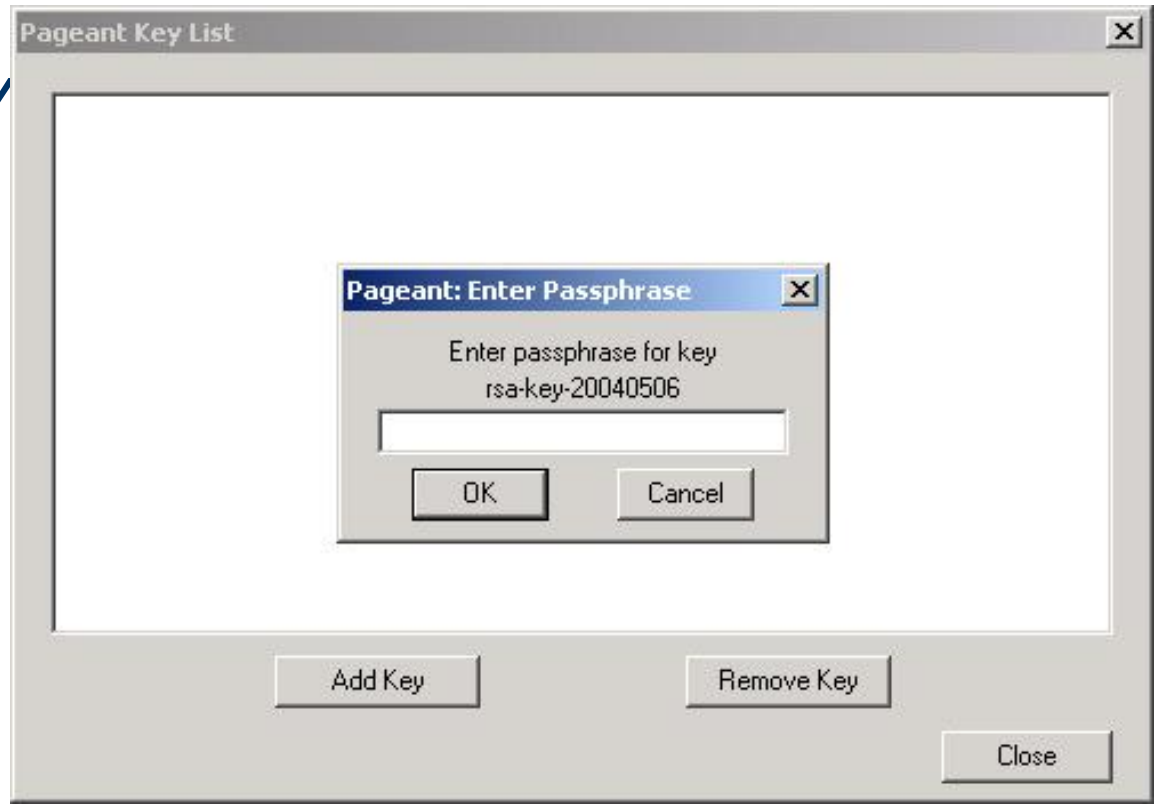
- Gestion des clés et agents avec PuTTY
  - Lancer l'agent en double cliquant sur l'icône ***pageant*** : 

  - 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

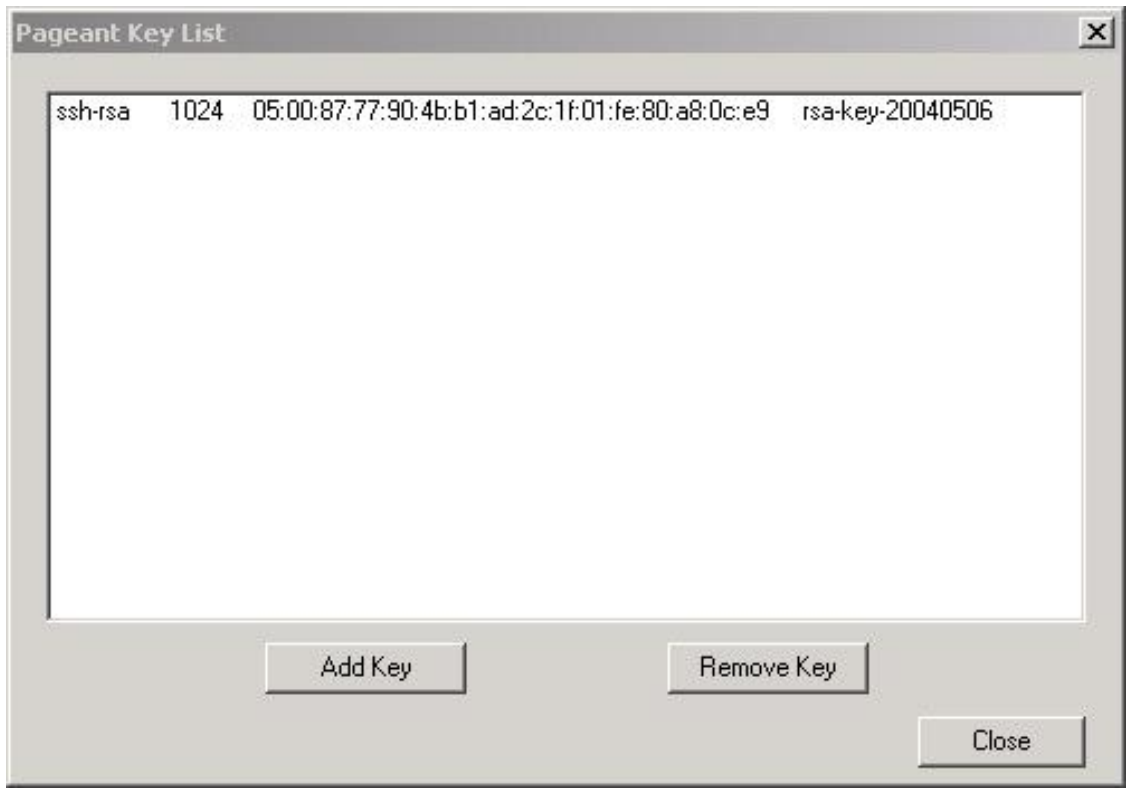
AF : ssh agent - PuTTY

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



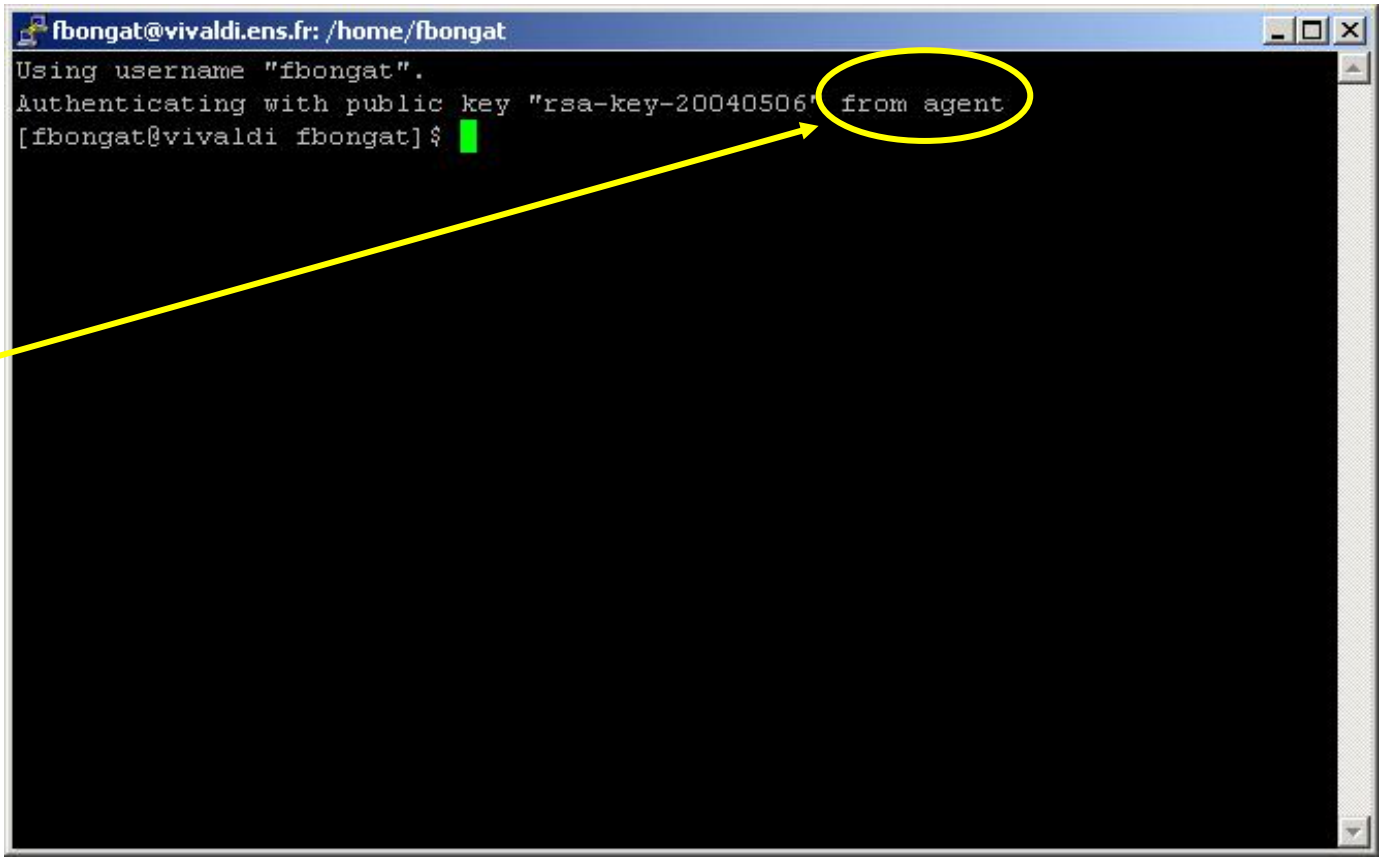
AF : ssh agent - PuTTY

- Gestion des clés et agents avec PuTTY
  - Les clés sont alors chargées dans l'agent SSH (en mémoire)
  - Il suffit alors de lancer une connexion PuTTY par authentification forte et aucun mot de passe ne sera plus demandé jusqu'à la fermeture de l'agent



AF : ssh agent - PuTTY

- Gestion des clés et agents avec PuTTY



Exemple de connexion ssh (valable aussi pour psftp et pscp) sans mot de passe qui est géré par l'agent

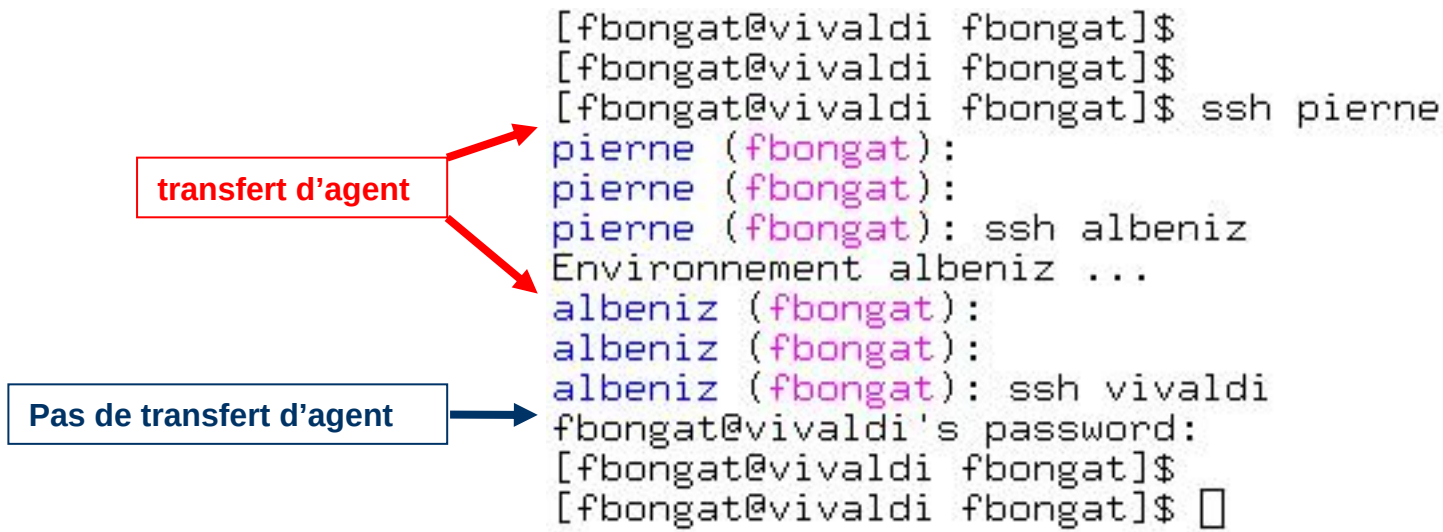
AF : transfert d'agent - Unix

- 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

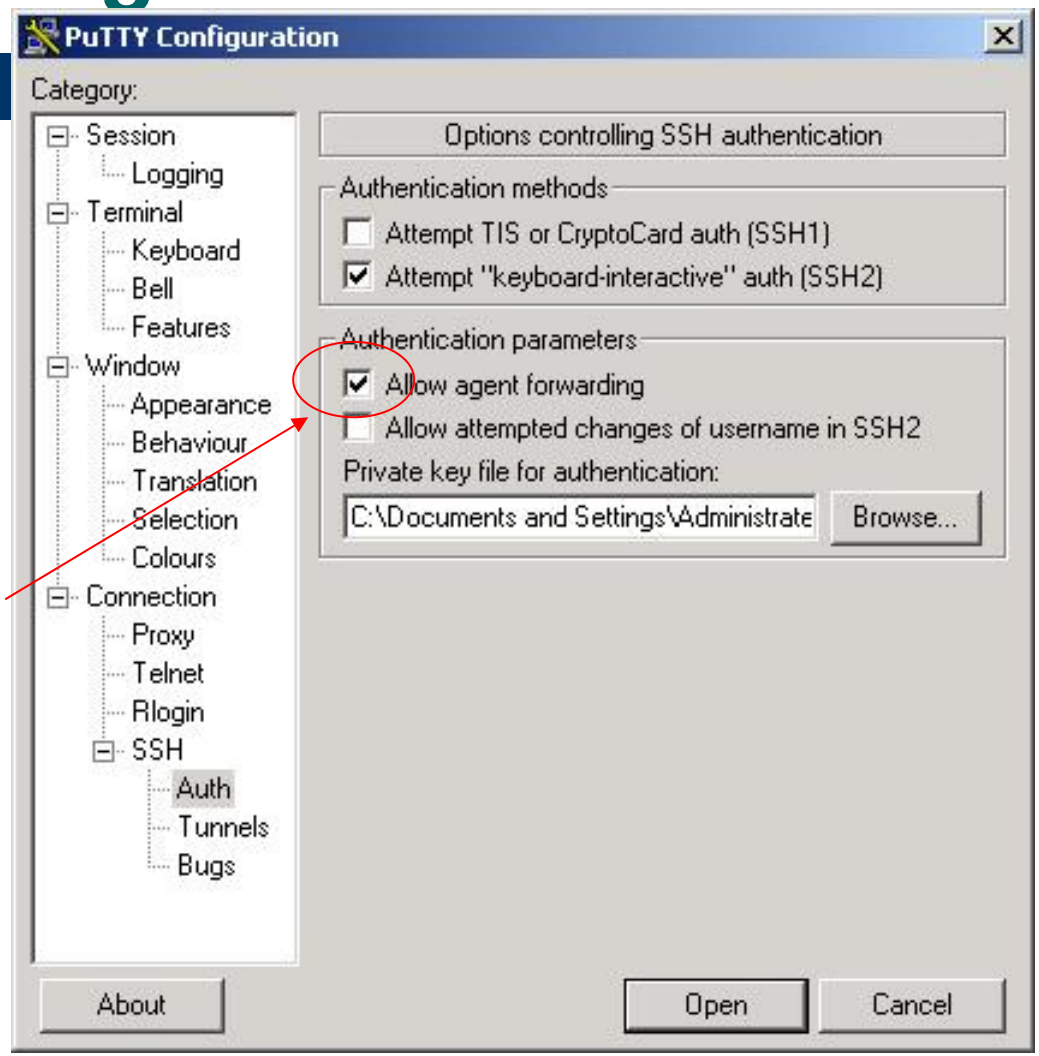
- Transfert d'agent
  - relais des demandes d'authentification entre hôtes



AF : transfert d'agent - PuTTY

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

      Allow agent forwarding



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

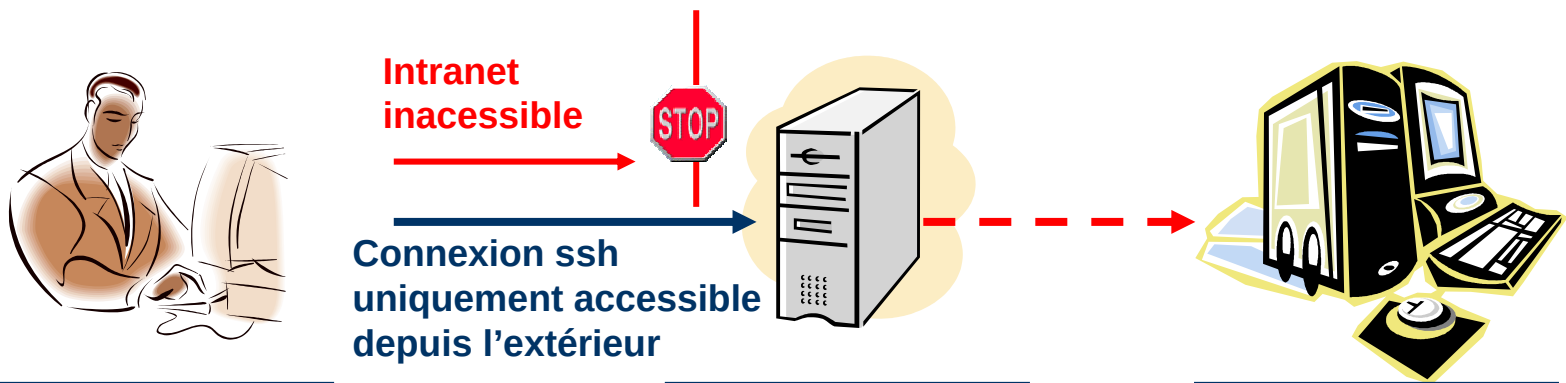
Tunneling : intranet

- Consultation d'un Intranet depuis l'extérieur

  Créer le tunnel sur la machine locale:

  ssh –n -f passerelle –L 18000:hote_B:80 sleep 180

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



Tunneling : messagerie Imap

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

  ssh –n –f –L 14300:localhost:143 machine2 sleep 180

  Possible aussi en V2 : ssh **–N** –f –L 14300:localhost:143 machine2
- Puis configurer l'application client imap sur localhost:14300
  - Soit nom du **serveur imap : localhost** et **port : 14300**

Mozilla-mail

Port local : 14300

machine1

Flux SSH

Serveur Imap Port : 143

Serveur - Port : 22

machine2

-n : redirection stdin /dev/null ***(pour les applications qui tournent en background)***

-f : mode background / sleep : durée de vie du tunnel sans connexion
-L : redirige le port local vers le port distant de l'hôte distant
-N : en V2, n'exécute pas de commande, remplace le sleep

Tunneling : messagerie Pop

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

    ssh –n –f –L 11000:localhost:110 machine2 sleep 180

    Possible aussi en V2 : ssh **–N** –f –L 11000:localhost:110 machine2
- Puis configurer l'application client pop sur localhost:11000
  - Soit nom du **serveur pop : localhost** et **port : 11000**

Mozilla-mail

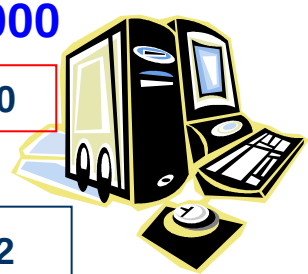
Port local : 11100

machine1

Flux SSH

Serveur POP Port : 110

Serveur - Port : 22



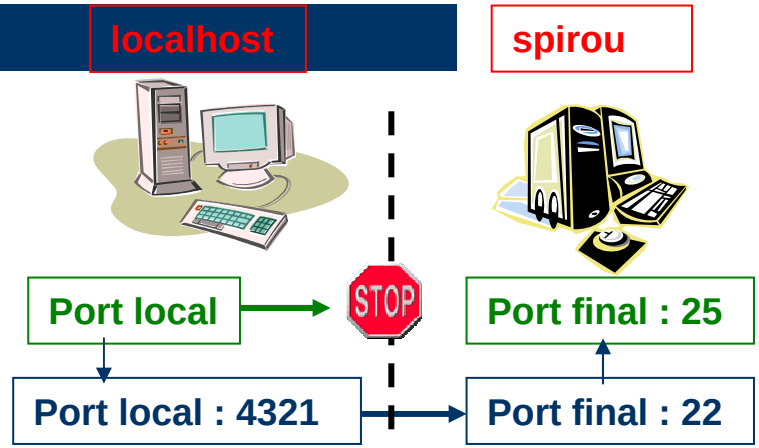
machine2

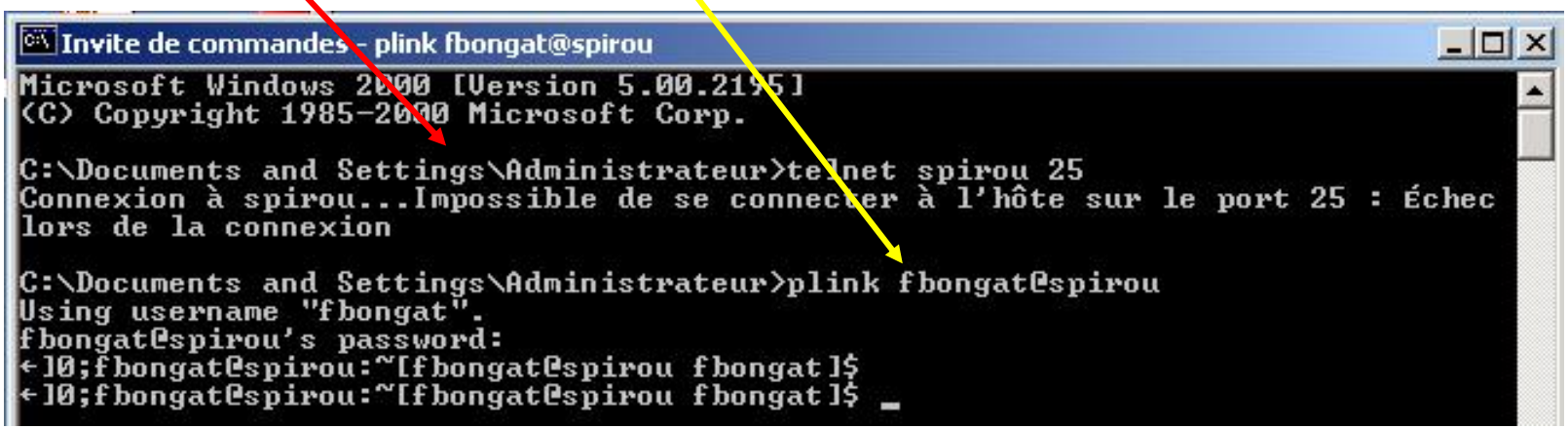
-n : redirection stdin /dev/null ***(pour les applications qui tournent en background)***

-f : mode background / sleep : durée de vie du tunnel sans connexion
-L : redirige le port local vers le port distant de l'hôte distant
-N : en V2, n'exécute pas de commande, remplace le sleep

Tunneling : envoi de messages

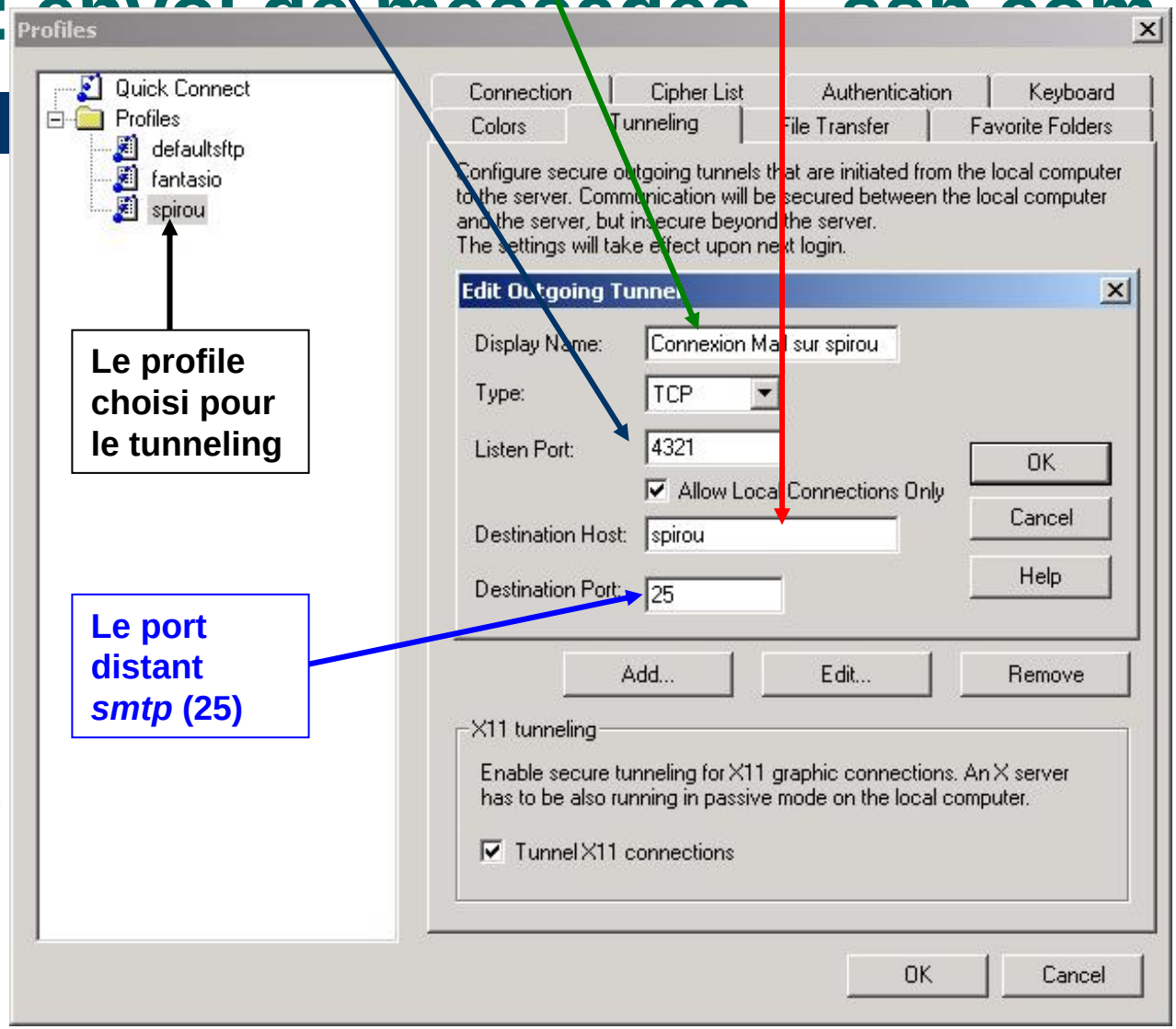
- Exemple d'une simulation d'un client de messagerie (telnet sur le port 25) vers un serveur smtp sortant
  - connexion sur le port 25 (smtp) de spirou qui est filtré
  - Seul le port 22 (ssh) est accessible depuis l'extérieur





Tunneling : envoi de messages ssh.com

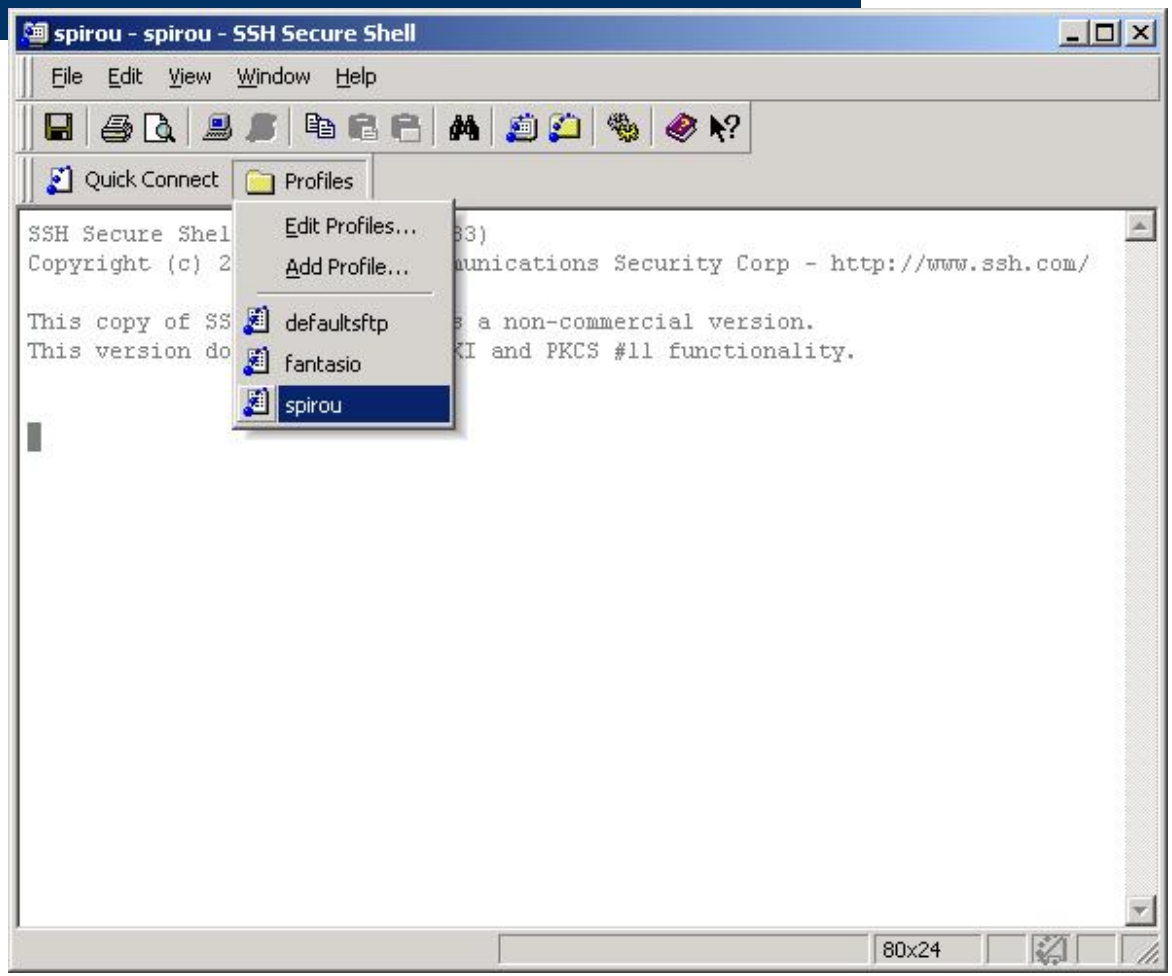
- ssh.com
  - Exemple smtp :
    - Ouvrir l'éditeur de profile, créer un profile pour la connexion spirou
    - Cliquer slur l'onglet ***Tunneling***
    - Choisir l'onglet ***Outgoing***
    - Puis sur ***Edit***
    - Remplir les champs liés au tunneling dans la fenêtre
    - ***Edit outgoing Tunneling***



Tunneling : envoi de messages - ssh.com

- ssh.com
  - Exemple smtp :
    - Lancer la connexion ssh avec le client

    - Cliquer sur l'icône ***Profiles*** puis ***spirou***
    - Entrer le mot de passe demandé afin d'obtenir une connexion classique
    - La connexion ssh établie, ***entraine*** le mise en place du Tunnel SSH



Tunneling : envoi de messages

- ssh.com
  - 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
_
```

Tunneling : en

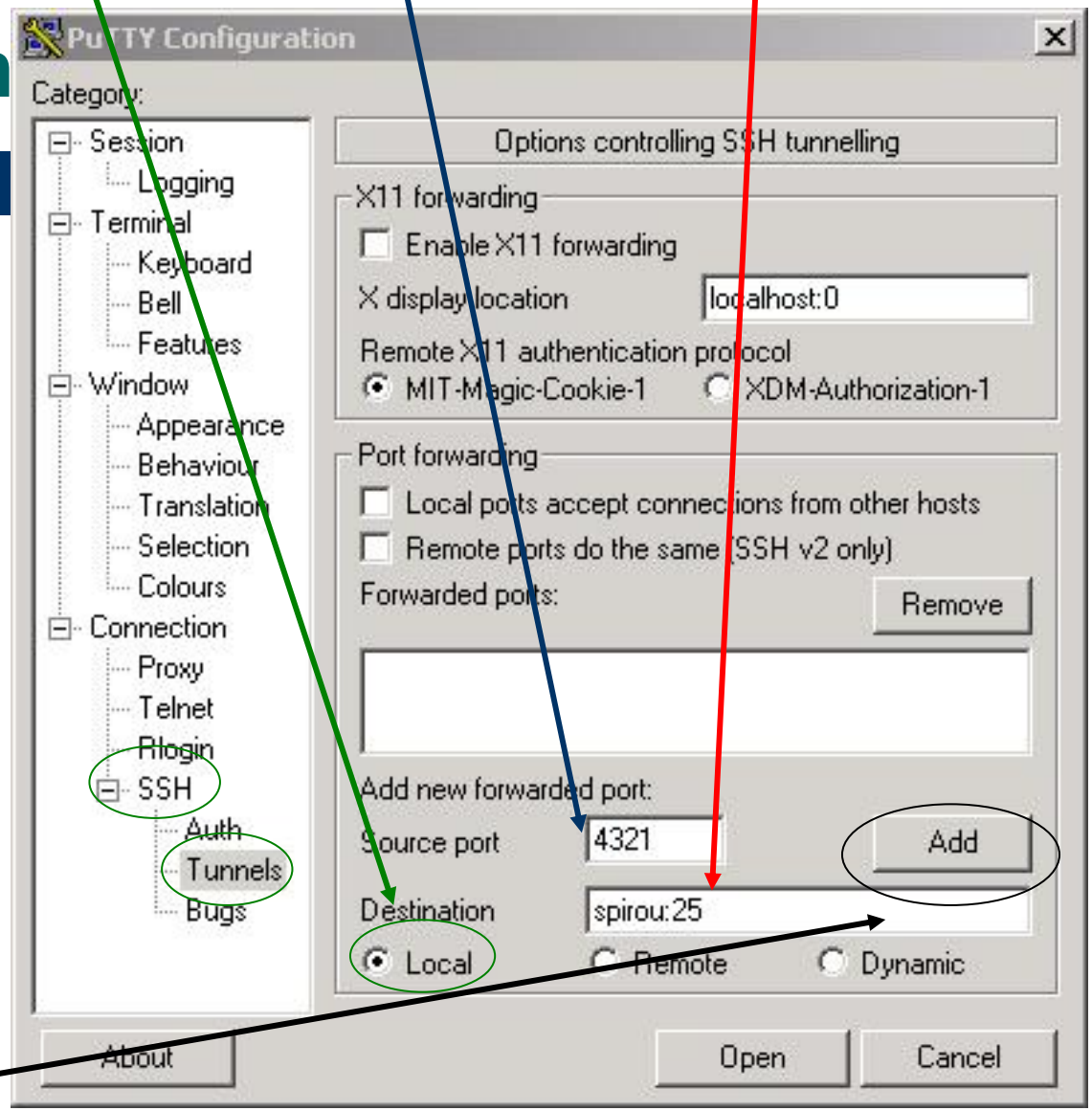
- PuTTY
  - Exemple smtp :
    - Lancer putty
    - Aller dans la valeur ***SSH*** → ***Tunnels***
    - Remplir les champs liés au tunneling dans la fenêtre
    - Puis valider cliquant sur ***Add***

Le port local (> 1024), ici : 4321

Connexion sur localhost

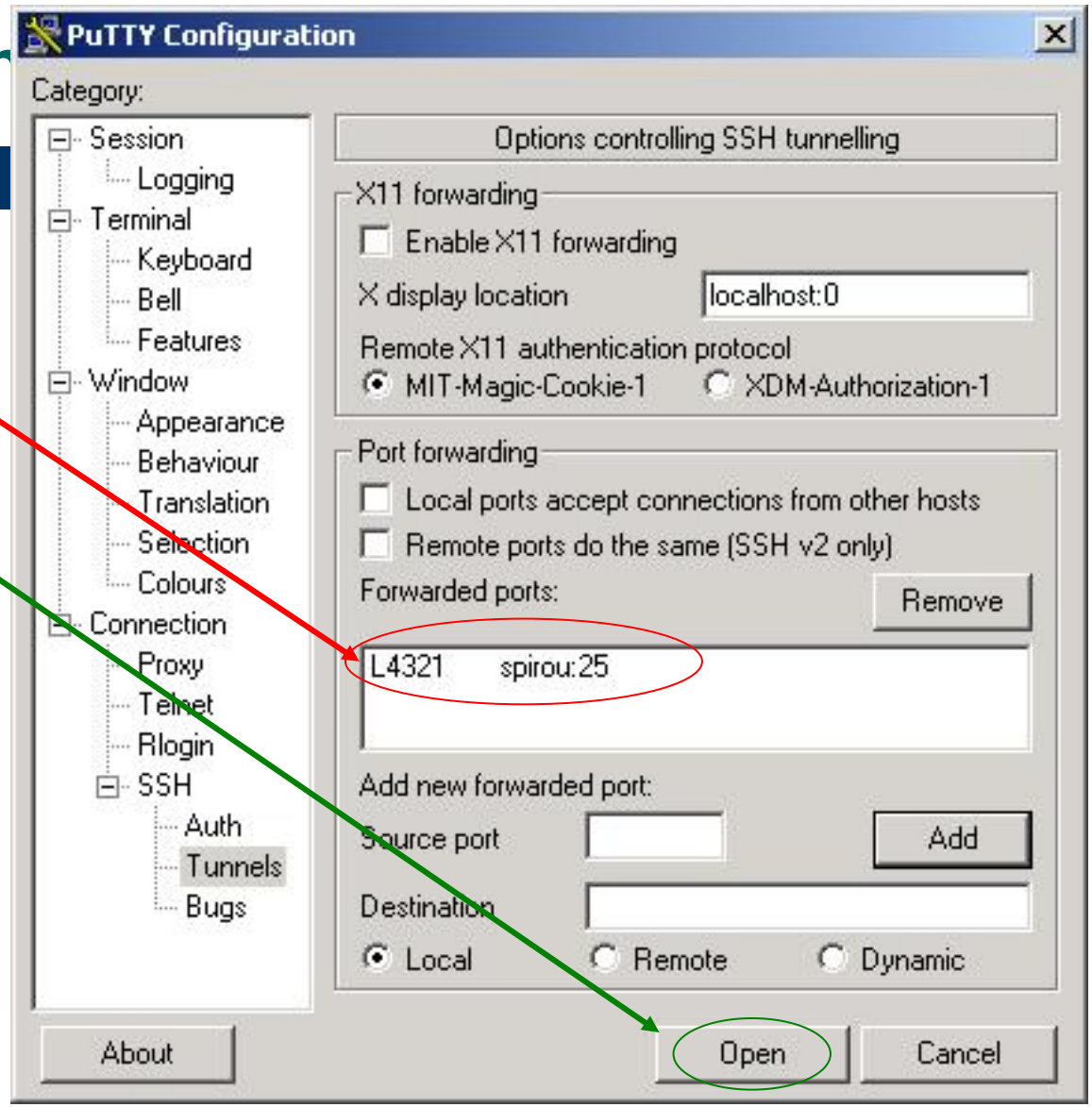
Nom de la machine distante et le port distant

Syntaxe : spirou:25



Tunneling : er

- PuTTY
  - Exemple smtp :
    - 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***



Conclusion

- SSH est une boîte à outils complète
- Les connexions sont sécurisées
- Existe différentes possibilités de connexions et de transferts de fichiers
- Les relais possibles des applications TCP sont un véritable plus de ssh
- Installé en standard sur tous les systèmes (client et serveur) Unix (ou presque) et MacOS X
- De très bons clients Windows : ssh.com et PuTTY

Remerciements à

- pour leurs relectures et leurs corrections :

  - Philippe Weill (Service Aéronomie)
  - Laurent Bourdette (Institut Pierre et Simon Laplace)
  - Robert Franchisseur (Laboratoire de Météorologie Dynamique)
  - Loïc Tortay (Institut National de Physique Nucléaire et de Physique des Particules)
  - Joël Marchand (Institut de Mathématique de Jussieu)
  - Michel Chabanne (Ecole Polytechnique – DSI)