Systèmes de fichiers distribués sécurisés

Pascal Véron
Groupe de Recherche en Informatique
et Mathématiques
Université de Toulon-Var

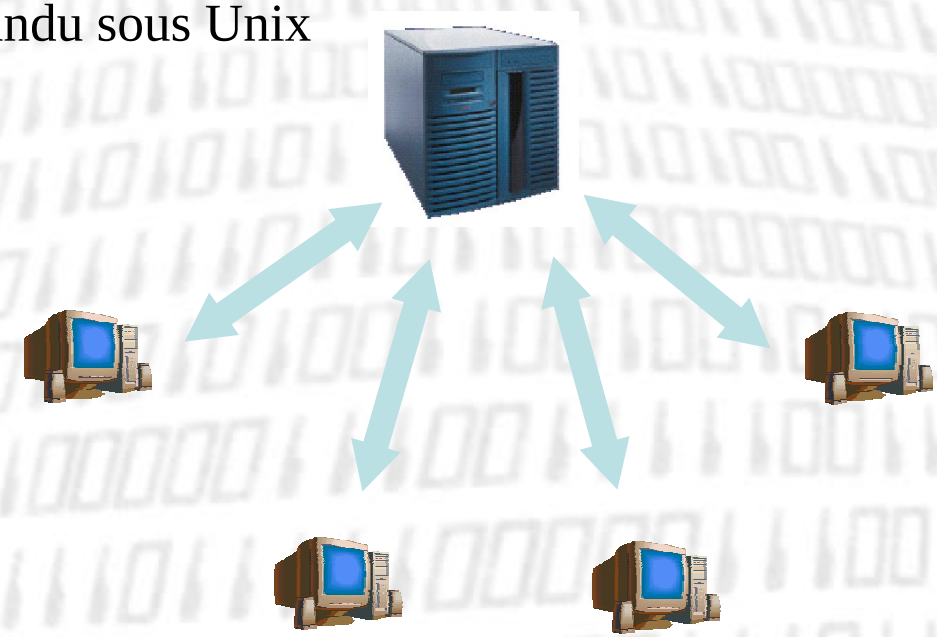
Le protocole NFS

Systèmes de fichiers distribués sécurisés

Pascal Véron
Groupe de Recherche en Informatique
et Mathématiques
Université de Toulon-Var

Le protocole NFS

- ✓ Développé dans les années 80 par Sun Microsystems
- ✓ Version courante: 3 (RFC 1813, juin 95)
- ✓ Permet la mise en place d'un espace de stockage centralisé
- ✓ Extrêmement répandu sous Unix

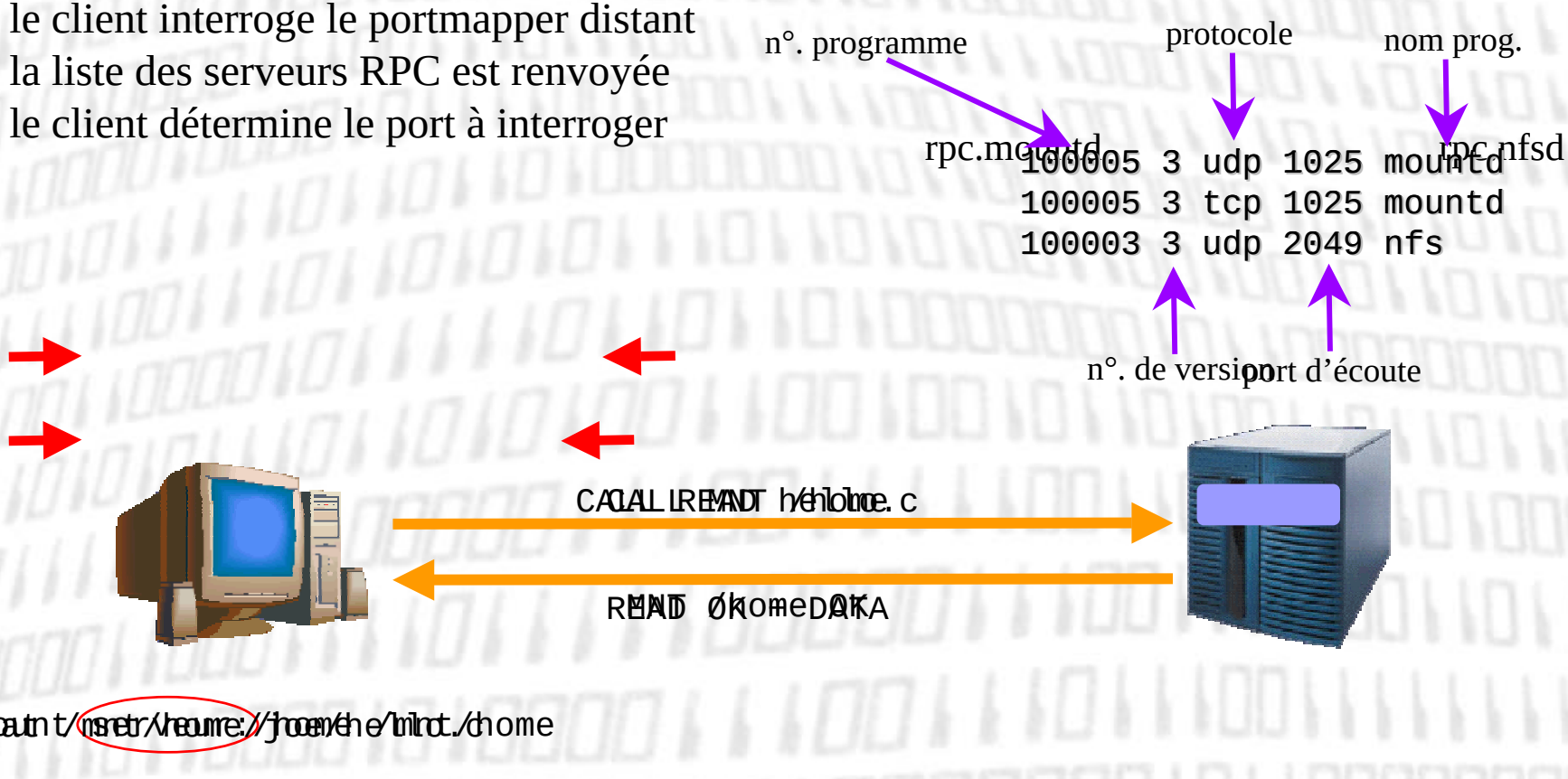


Le protocole NFS

- ✓ NFS est constitué de 2 protocoles:
  - Mount: permet d'obtenir un descripteur de fichiers sur le système exporté
  - Nfs: traite les demandes d'accès (lecture, écriture)
- ✓ Le protocole utilise le mécanisme RPC (Remote Procedure Call) pour l'échange des messages entre les clients et le serveur.

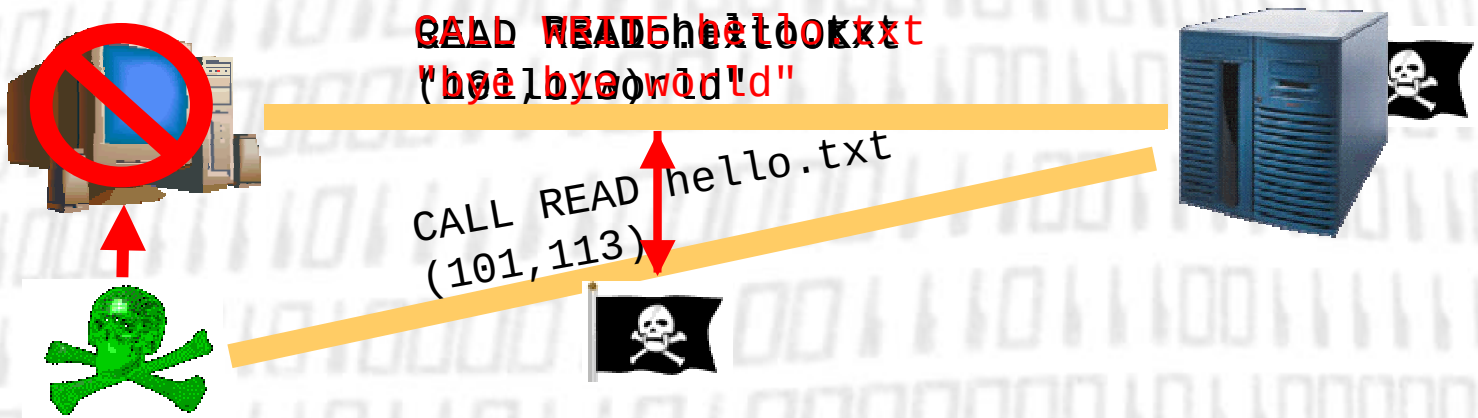
RPC

- ✓ Protocole permettant à un programme « d 'exécuter une fonction » d'un programme distant.
- ✓ Principe de fonctionnement:
  - les serveurs RPC s'enregistrent auprès du portmapper local
  - le client interroge le portmapper distant
  - la liste des serveurs RPC est renvoyée
  - le client détermine le port à interroger



Faiblesses du protocole NFS

- ✓ Le serveur n'est pas authentifié
- ✓ Le client n'est pas authentifié
- ✓ La validité des requêtes RPC est assurée par le couple (uid, gid) associée
- ✓ Les données transitent en clair sur le réseau
- ✓L'intégrité des données échangées n'est pas vérifiée.



SSH

✓ Développé en 1995 par T. Ylönen

✓ Composé de 3 couches:

• la couche transport: authentification du serveur, confidentialité et intégrité des données échangées.

• la couche authentification: authentification de l'utilisateur par le serveur.

• la couche connexion: sessions interactives, exécution de commande distante, gestion du flux X11, transmission des variables d'environnement … et

élaboration de tunnels sécurisés (mécanisme de redirection de ports)

La redirection de ports avec SSH

✓Déclaration d'un tunnel SSH:

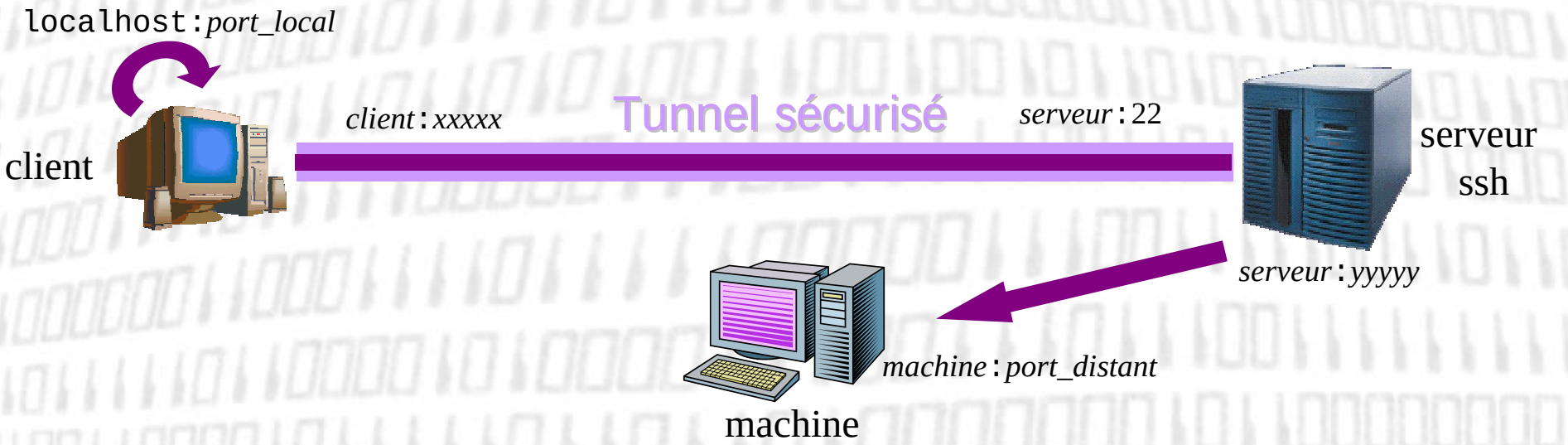
`ssh -L` *port_local:machine:port_distant* [*login@*]*serveur_ssh*

✓ Mise en place d'une socket tcp sécurisée entre le port local du poste client et le port 22 du serveur SSH.

✓ Le tunnel n'est accessible que via l'interface réseau loopback du poste client.

✓ Toute requête vers `localhost`:*port_local* est transmise à *machine:port_distant* via le serveur SSH.

✓ Pour le poste distant, la requête provient du serveur SSH.



La redirection de ports avec SSH

✓ 2 méthodes de création:

- tunnel permanent

```
ssh -f -N -L port_local:machine:port_distant [login@]serveur_ssh
```

- tunnel temporaire

```
ssh -f -L port_local:machine:port_distant [login@]serveur_ssh sleep 30
```

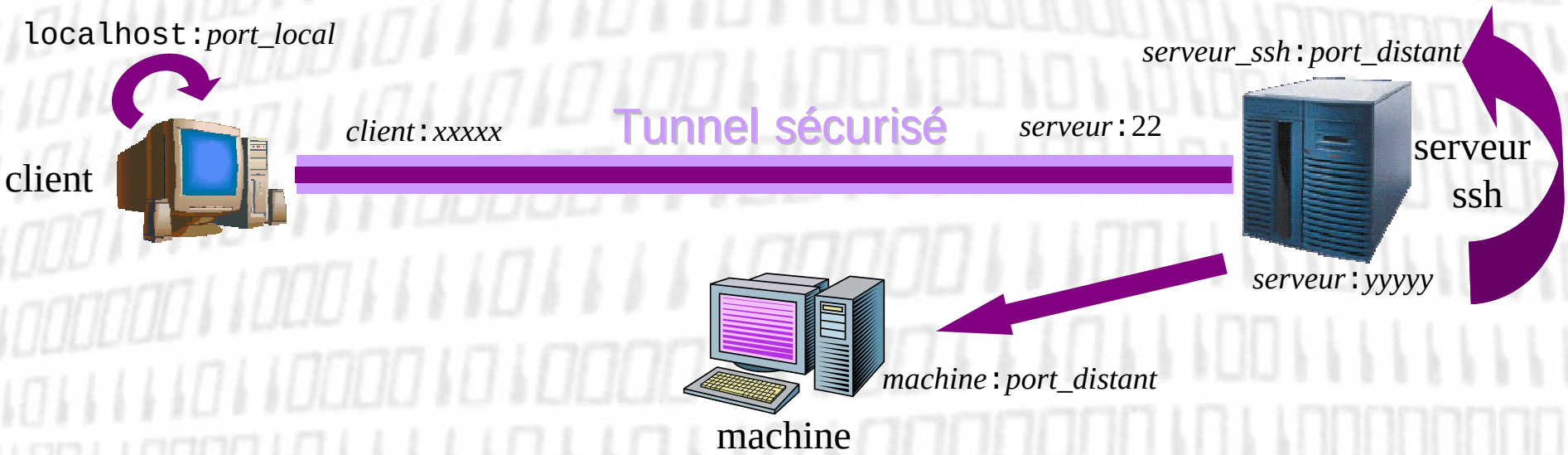
-f : ne pas rester en tâche de fond après la passphrase saisie

-N : pas d'exécution de commande distante

Au-delà de 30 secondes, le tunnel reste actif si une application l'utilise.

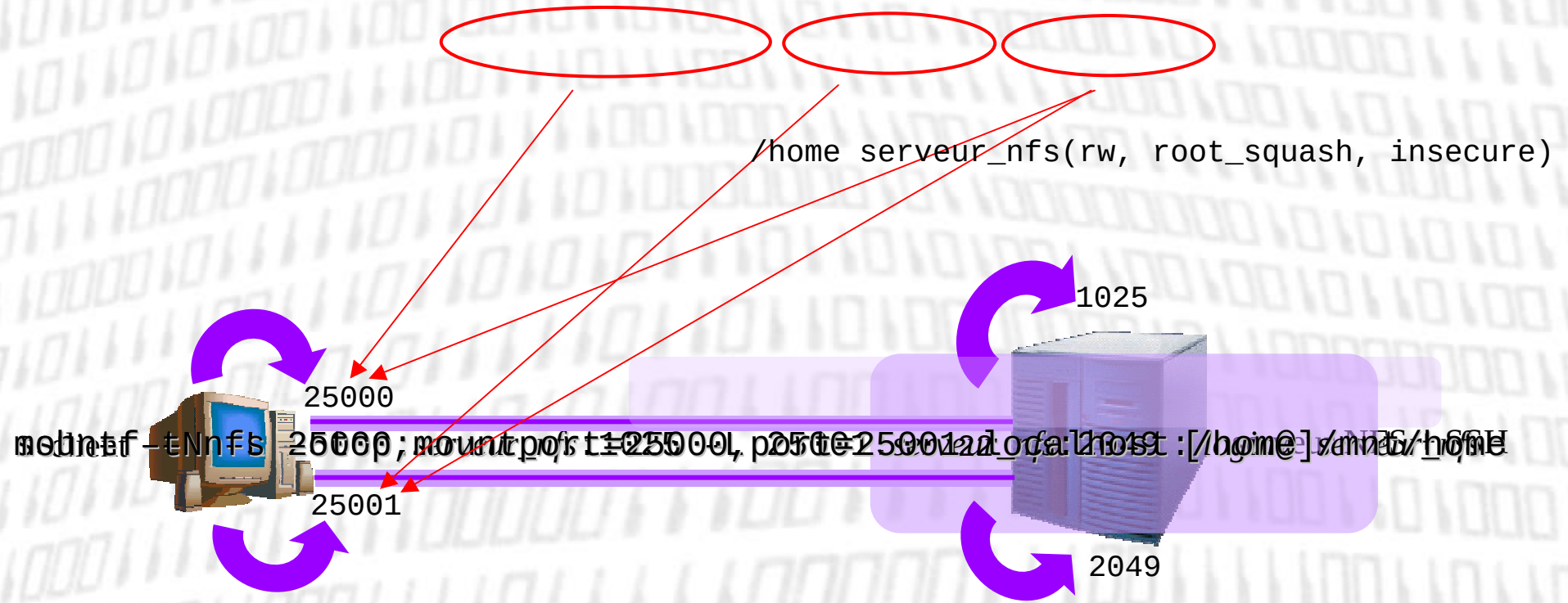
✓ Cas particulier:

```
ssh -f -L port_local:machine:port_distant [login@]serveur_ssh sleep 30
```



Sécuriser NFS avec SSH (cas d'un serveur TCP)

- ✓ Déterminer les ports d'écoute des démons *mount* et *nfs*
- ✓ Créer sur le poste client le tunnel SSH
- ✓ Monter la partition exportée via le tunnel
- ✓ Sans oublier de configurer le serveur …



Sécuriser NFS avec SSH (cas d'un serveur TCP)

✓ Automatisation de la procédure:

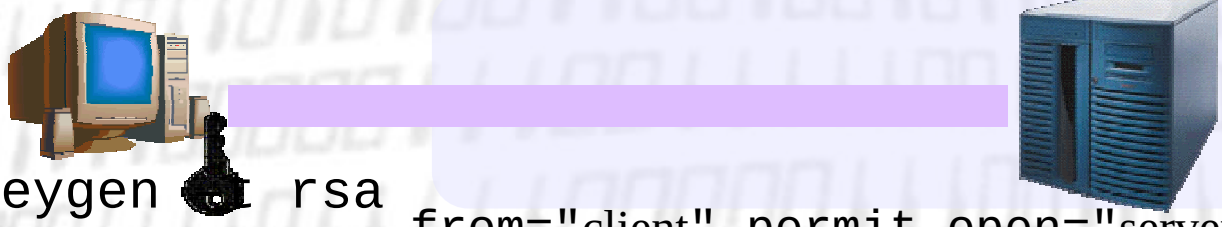
- **`/etc/fstab`**

```
localhost:/home /mnt/home nfs,tcp,intr,bg,mountport=25000,port=25001
```

- Authentification automatique:

créer sur le poste client un couple de clés sans passphrase,
placer la clé publique sur le serveur: `~login/.ssh/authorized_keys`

```
ssh -f -i ~/.ssh/id_rsa -L 25000:serveur_nfs:1025 -L 25001:serveur_nfs:2049 [login@]serveur_nfs sleep 30
```



```
ssh-keygen -t rsa
```

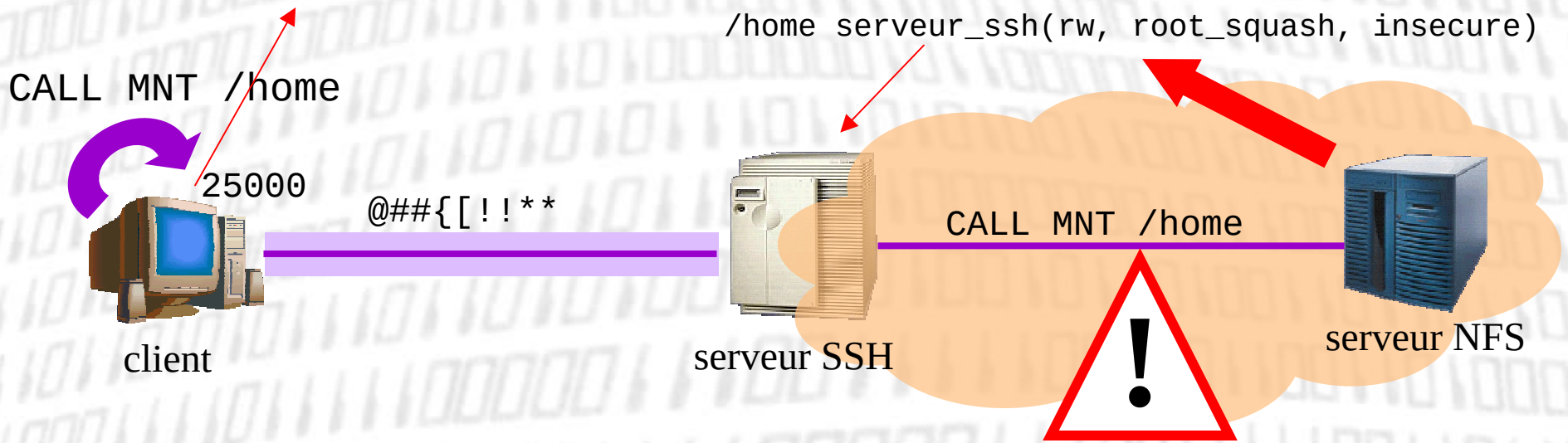
```
from="client",permit-open="serveur_ssh:1025",
permit-open="serveur_ssh:2049",
command="/bin/sleep 300",
ssh-rsa AAAAB3NzaC1yc2EA……root@client
```

Sécuriser NFS avec SSH (cas d'un réseau privé)

- ✓ Créer un tunnel via le serveur SSH
- ✓ Configurer le serveur NFS

```
ssh -f -L 25000:serveur_nfs:1025
       -L 25001:serveur_nfs:2049 [login@]serveur_ssh sleep 30
```

```
mount -t nfs -otcp,mountport=25000,port=25001 localhost:/home /mnt/home
```



les communications ne sont pas chiffrées entre le serveur nfs et le serveur ssh.

Sécuriser NFS avec SSH (cas d'un réseau privé)

- ✓ Créer un tunnel du client vers le serveur SSH, puis du serveur SSH vers le serveur NFS.

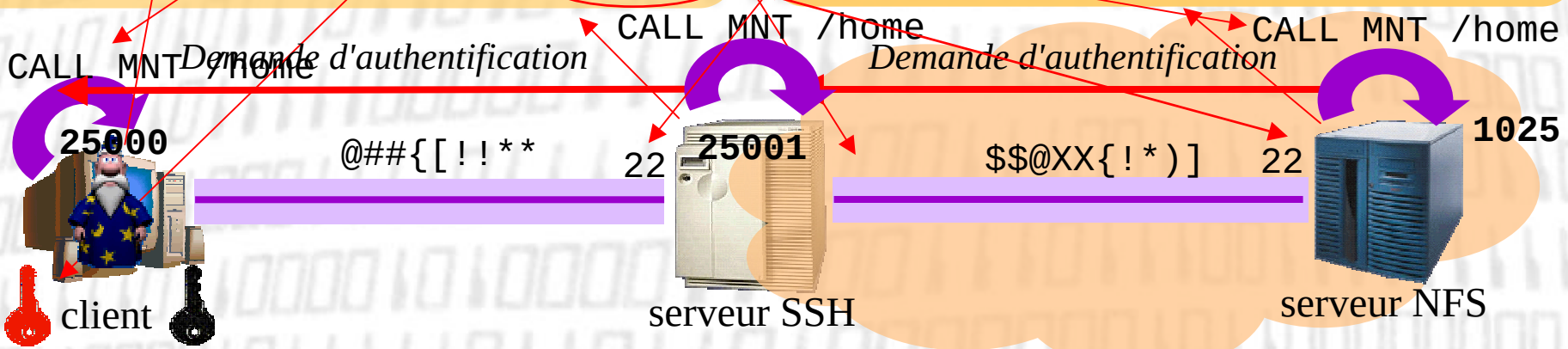
```
eval `ssh-agent`; ssh-add id_rsa_tunnel
ssh -f -A -L 25000:localhost:25001
          -L 25002:localhost:25003 [login@]serveur_nfs sleep 30
```

- ✓ Comment gérer l'authentification ? Utilisation de l'agent SSH

```
mount -t nfs -otcp,mountport=25000,port=25002 localhost:/home /mnt/home
```

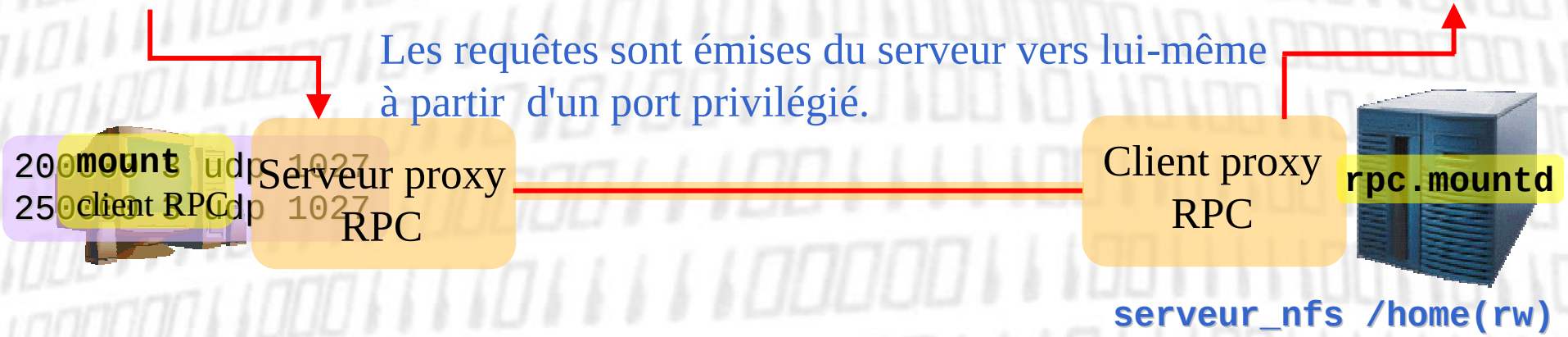
```
from="client",
permit-open="localhost:25001",
permit-open="localhost:25003",
command="ssh -T -L 25001:serveur_nfs:1025
-L 25003:serveur_nfs:2049 [login@]serveur_nfs"
```

```
from="serveur_ssh",
permit-open="serveur_nfs:1025",
permit-open="serveur_nfs:2049",
command="/bin/sleep 300"
```



SEC RPC (Serveur NFS UDP)

- ✓ Développé par J. Bowmman, version 1.52-1, 07/03
- ✓ Composé essentiellement de deux programmes:
  - `rpc_psrv`: serveur proxy RPC à placer sur le client
    lance automatiquement via SSH (id utilisateur: snfs)
  - `rpc_pcl`: client proxy RPC à placer sur le serveur (suid root)
- ✓ Les requêtes UDP sont encapsulées dans une trame TCP
- ✓ Le client proxy se charge de les transmettre aux serveurs RPC




```
mount -t nfs -omountprog=200000,nfsprog=250000 client:/home /mnt/home
```

SHFS

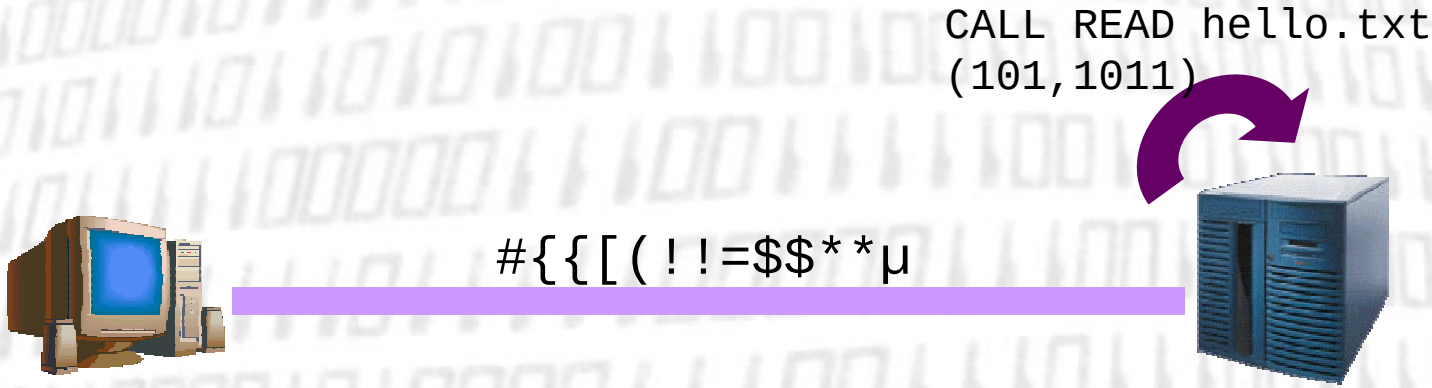
- ✓ Développé par M. Spousta, version 0.32, 5/11/03
- ✓ Ne nécessite pas de serveur NFS … mais uniquement un serveur SSH
- ✓ Consiste en un module du noyau à installer sur le poste client: `shfs.o`

- ✓ Principe de fonctionnement:
  - Ouverture d'un shell SSH vers le serveur
  - Toute requête système vers la partition distante est transformée en une requête shell
  - Pas de possibilité de monter la partition utilisateur au boot
  - Permet une authentification de l'utilisateur

Bilan sur l'utilisation de SSH

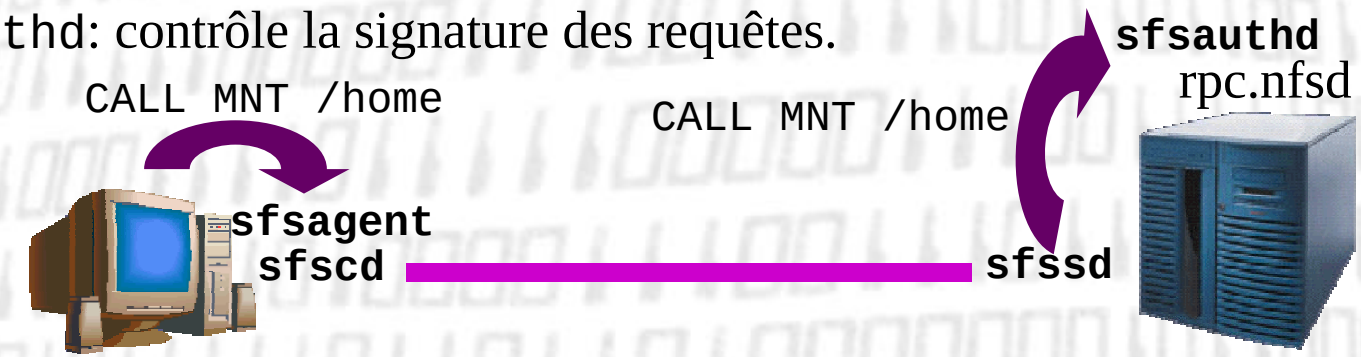
- ✓ ~~Le serveur n'est pas authentifié~~
- ✓ Le client n'est pas authentifié — le créateur du tunnel est authentifié **sauf pour shfs**
- ✓ La validité des requêtes RPC est assurée par le couple (uid, gid) associée
- ✓ ~~Les données transitent en clair sur le réseau~~
- ✓ ~~L'intégrité des données échangées n'est pas vérifiée~~.

→ *sauf sur l'interface locale*



SFS (Self-certifying File System)

- ✓ Développé par D. Mazières en Mai 2000 au (Massachussets Institute of Technology)
- ✓ Version actuelle: 0.7.2-1, Mai 2003.
- ✓ SFS assure:
  - l'authentification du serveur par le poste client
  - l'authentification du poste client par le serveur
  - le chiffrement et l'intégrité des données échangées sur le réseau
  - une authentification à base de signature numérique de chaque requête RPC émise vers le serveur.
- ✓ SFS se compose essentiellement de 4 programmes:
  - `sfscd, sfssd`: gestion des communications client-serveur
  - `sfsagent`: signe les requêtes nfs
  - `sfsauthd`: contrôle la signature des requêtes.



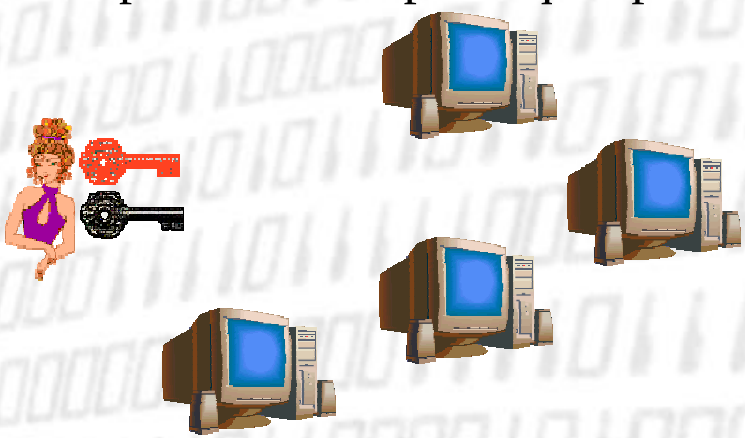
SFS (Self-certifying File System)

- ✓ Chaque serveur possède un couple de clés privée/publique pour l'authentification et l'établissement d'une clé de session
- ✓ Les données sont accessibles à partir du client via le chemin:

`/sfs/`*@serveur_nfs,HOSTID*

où *HOSTID* = SHA-1(1,*serveur_nfs*, 🔑,1,*serveur_nfs*,🔑)

- ✓ Comment obtenir *HOSTID* ?
- ✓ Le protocole a été développé afin qu'un utilisateur puisse accéder au serveur à partir de n'importe quel poste sans avoir à dupliquer sa clé privée.

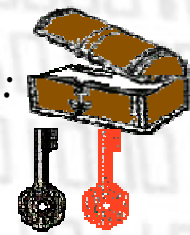


`/sfs/`*@serveur_nfs,HOSTID*

SFS: Un exemple d'utilisation

- ✓ Enregistrement de l'utilisateur sur le serveur
- ✓ La clé privée est protégée sur le serveur par une passphrase 
- ✓ Accès aux données:

login:uid:gid:SRP data:  *:*

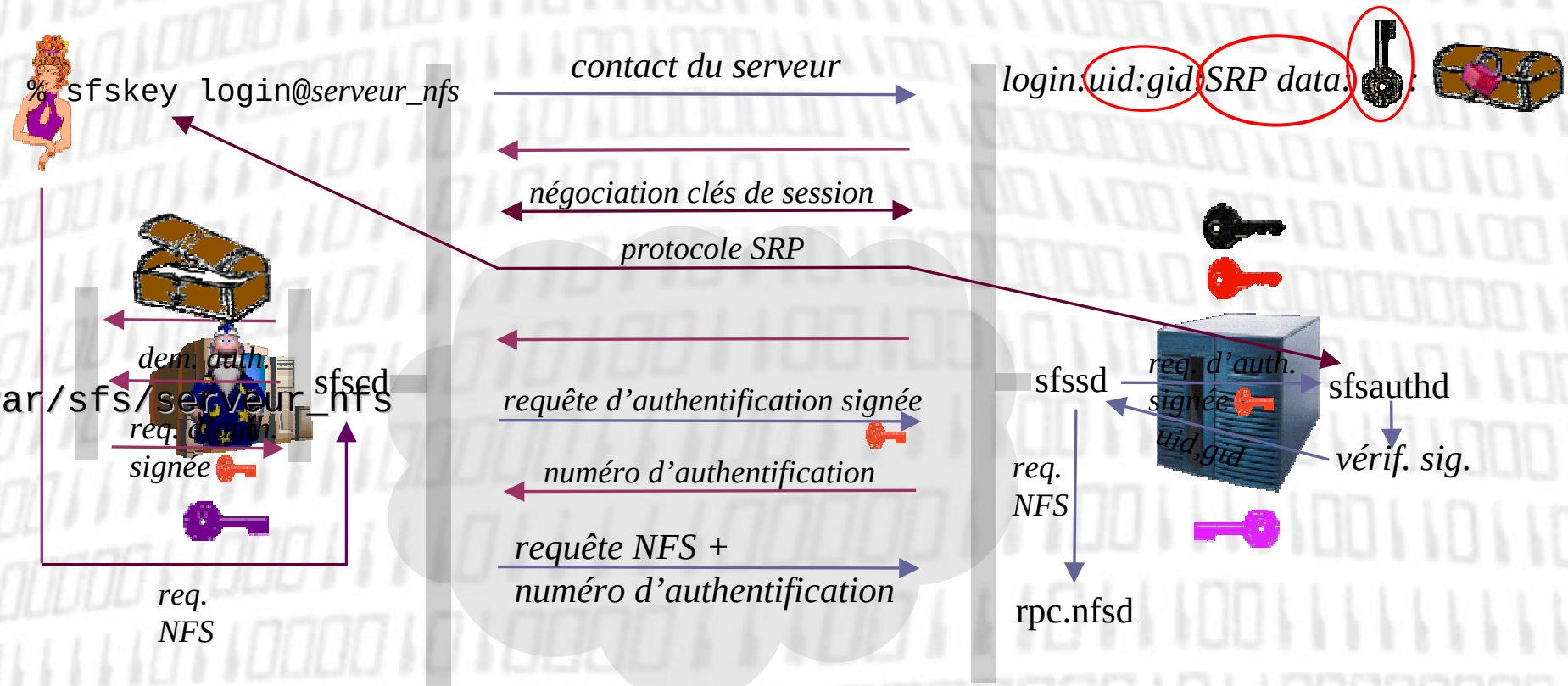


```
%sfskey register
```

SFS: Un exemple d'utilisation

- ✓ Enregistrement de l'utilisateur sur le serveur
- ✓ La clé privée est protégée sur le serveur par une passphrase

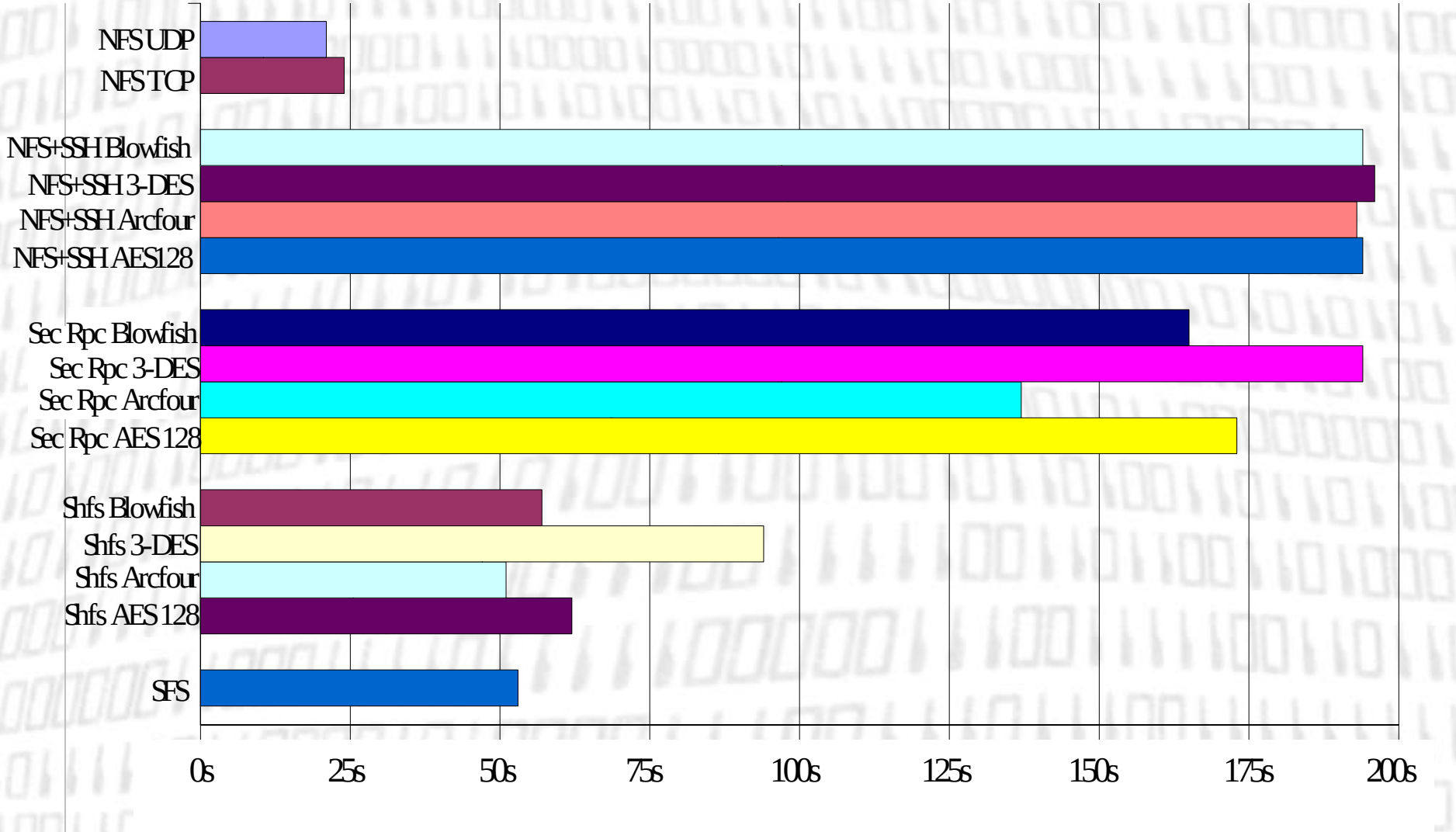
- ✓ Accès aux données:



Analyse des performances

PIII 1Ghz  *100 Mo*  PII 350Mhz

100 Mbits/sec



NFS 4

- ✓ RFC 3530, Avril 2003
- ✓ Nouveau mécanisme d'authentification pour les requêtes RPC: **RPCSEC_GSS**
- ✓ Repose sur la GSSAPI: interface de programmation générique encapsulant plusieurs mécanismes de sécurité
- ✓ Déroulement d'une session de type **RPCSEC_GSS**:
  - Création du contexte: négociation du mécanisme de sécurité et du type de service à mettre en place
    - service d'authentification
    - service d'intégrité
    - service de confidentialité
  - Echange des données
  - Destruction du contexte
- ✓ Mécanismes de sécurité disponibles:
  - ~~KERBEROS V5~~
  - LIPKEY: MD5, RSA, CAST
  - SPKM-3

And the winner is …