# *La mobilité : quelles solutions ?*

## *Séminaire RAISIN 24 mars 2005*

Roland Dirlewanger
CNRS -Délégation Aquitaine-Limousin
Esplanade des Arts et Métiers
33402 Talence Cedex

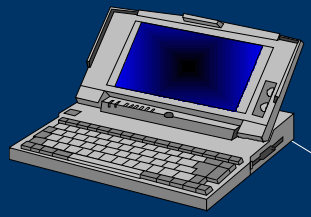
Roland.Dirlewanger@dr15.cnrs.fr

# *Les enjeux*

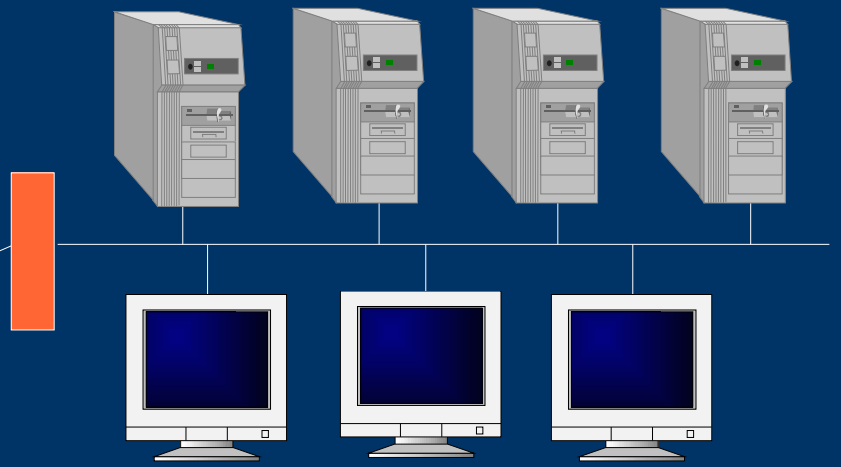
- Le point de vue de l'utilisateur
  - On peut se raccorder à l'Internet depuis n'importe où
  - Je peux donc accéder aux ressources informatiques de mon unité d'où je veux, quand je veux
- Le point de vue de l'administrateur systèmes et réseaux
  - Je dois protéger ces ressources informatiques
- Comment concilier les deux ?

# Les différents cas de figure *(1)*

Connexion sur un réseau tiers
- salle libre service d'une conférence
- visite dans un autre labo



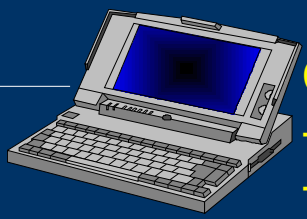
Internet



Réseau de l'unité

Connexion à domicile
- ADSL
- Téléphone

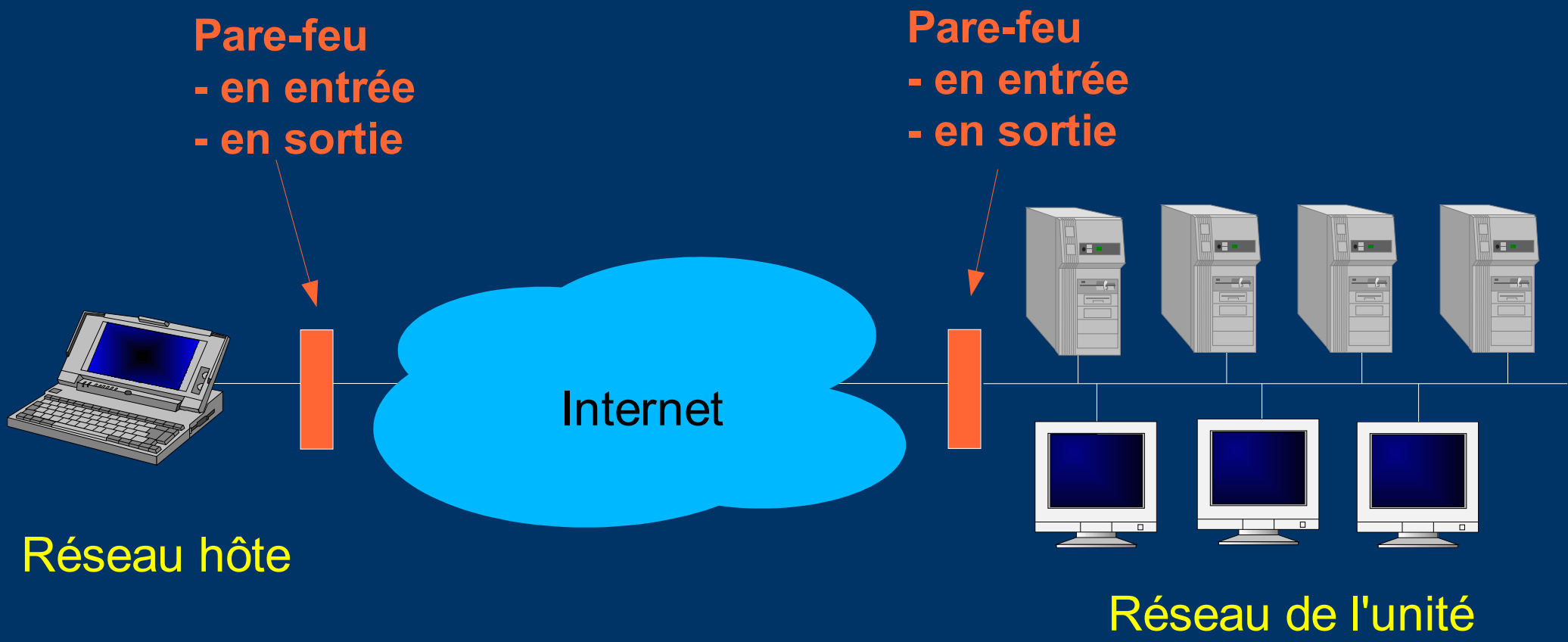
Utilisation d'un poste tiers
- cybercafé
- dans un autre labo



Connexion sans fil
- WiFi
- via téléphone portable

# *Les différents cas de figure (2)*

- Ces différents cas de figure se ramènent à :



# *Les besoins*

- Accès à la messagerie : lecture, envoi
- Accès intranet WWW
- Accès aux serveurs de fichiers
- Accès à des bases de données
- Accès à des applications
- Accès à une bulle protégée dans un réseau “hostile”
- ...

# *Les contraintes*

- Eviter la multiplication des points d'entrée dans le réseau du site qui oblige à :
  - Maintenir à jour un grand nombre de serveurs
  - Maintenir un inventaire rigoureux de ce qui est ouvert

- Eviter la transmission en clair d'informations d'authentification
  - L'accès à une ressource peu sensible se fait bien souvent avec le même mot de passe que pour une ressource plus sensible !

# *Solutions pour la messagerie (1)*

- Webmail via un serveur HTTPS
  - Nécessite un certificat pour le serveur HTTP
  - Permet d'utiliser les ressources informatiques d'un tiers
  - Confidentialité des messages
  - Protection des mots de passe

  - Confiance dans le poste de travail  utilisé ?
  - Mémorisation des mots de passe ?

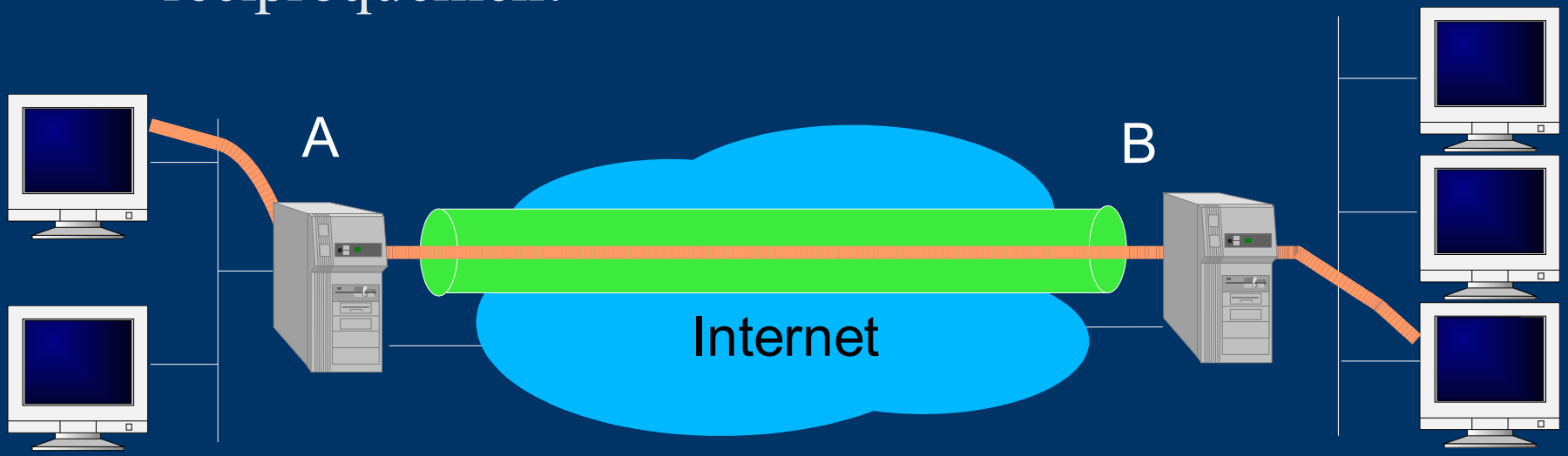
# *Solutions pour la messagerie (2)*

- Clients SSL : IMAP/S, SMTP/TLS
  - Nécessite un poste de travail préconfiguré et un degré d'expertise de l'utilisateur pour le connecter au réseau hôte
  - Recommandé : authentification de l'utilisateur via un certificat personnel => inutile de reconfigurer la messagerie pour éviter les problèmes d'anti-relais

  - Résistance du poste de travail aux attaques depuis le réseau hôte ?
  - Tendance à fermer le port 25 en sortie ?

# Les tunnels

- Idée générale :
  - Canal authentifié et éventuellement chiffré entre deux points A et B
  - Achemine du trafic de l'amont de A vers l'aval de B et réciproquement

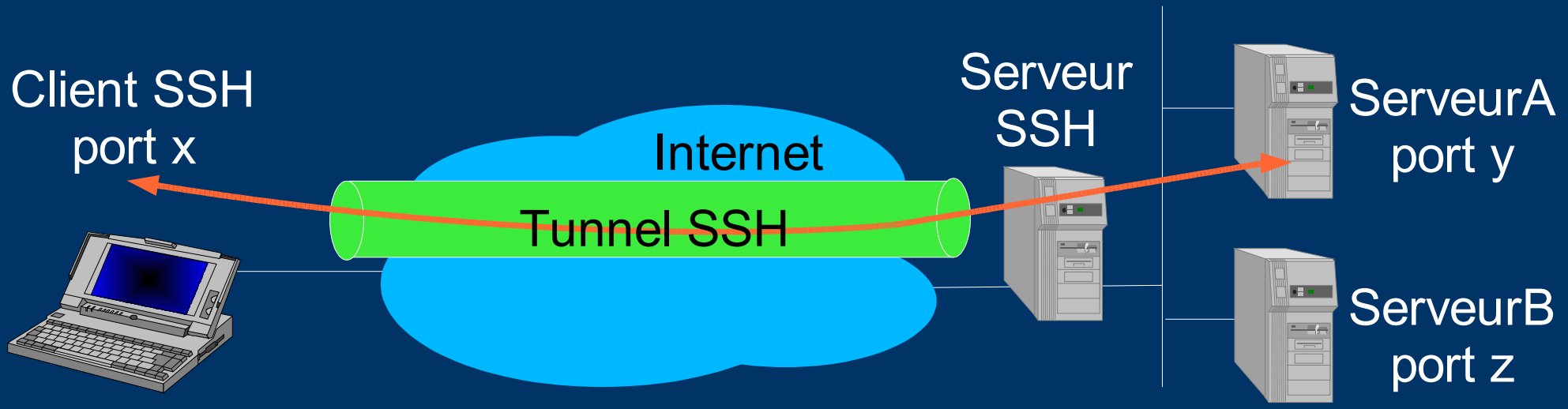


# *Les tunnels SSH (1)*

- SSH ou Secure shell
  - A la fois : un protocole et l'implémentation de clients et de serveurs
  - Mis au point par Tatu Ylönen
    - version commerciale : www.ssh.fi
    - version libre : www.openssh.org
  - Alternative sécurisée à telnet, rlogin, rsh, etc.
    - Authentification par mot de passe ou par clé publique
    - Chiffrement
    - Réacheminement de ports (*port forwarding*)
  - Clients
    - Unix : ssh, scp, ...
    - Windows : putty, winscp, ...

# Les tunnels SSH (2)

- Le réacheminement de ports
  - Le client se connecte à un serveur
  - Il définit les réacheminements : par exemple 127.0.0.1:*x* est réacheminé vers *serveurA:y*
  - Toute connexion sur le client sur *127.0.0.1:x* est envoyée au serveur SSH qui établit la connexion avec *serveurA:y*



# *Les tunnels SSH (3)*

- Quelle authentification choisir ?
  - Par mot de passe :
    - Peut-être utilisé depuis un poste de travail tiers
    - SSH protège le mot de passe. D'autres protocoles ne le font pas => fuite de mots de passe, rejeu possible contre SSH !
  - Par clé privée/clé publique
    - Nécessite de stocker la clé privée (sur le poste de travail, sur une clé USB, etc.) et de la protéger par un mot de passe
    - L'utilisateur peut désactiver la protection par mot de passe !
- Un compromis :
  - Accès par clé privée/clé publique depuis l'extérieur
  - Accès par mot de passe ou clé privée/clé publique en interne

# *Les tunnels SSH (4)*

- Avantages :
  - Grande intéropérabilité (Unix, Linux, Windows, Mac, ...)
  - Grande simplicité de configuration
  - Réacheminement de X11 (plus besoin de "`xhost +`*xyz*")
- Inconvénients :
  - Certains services ne sont pas tunnelisables :
    - UDP
    - SMB sur client Windows (mais on peut utiliser WinSCP)
    - HTTP 1 .1 (mais on peut configurer un proxy)
  - Pas vraiment transparent pour l'utilisateur
    - Il doit configurer à l'avance les tunnels dont il aura besoin

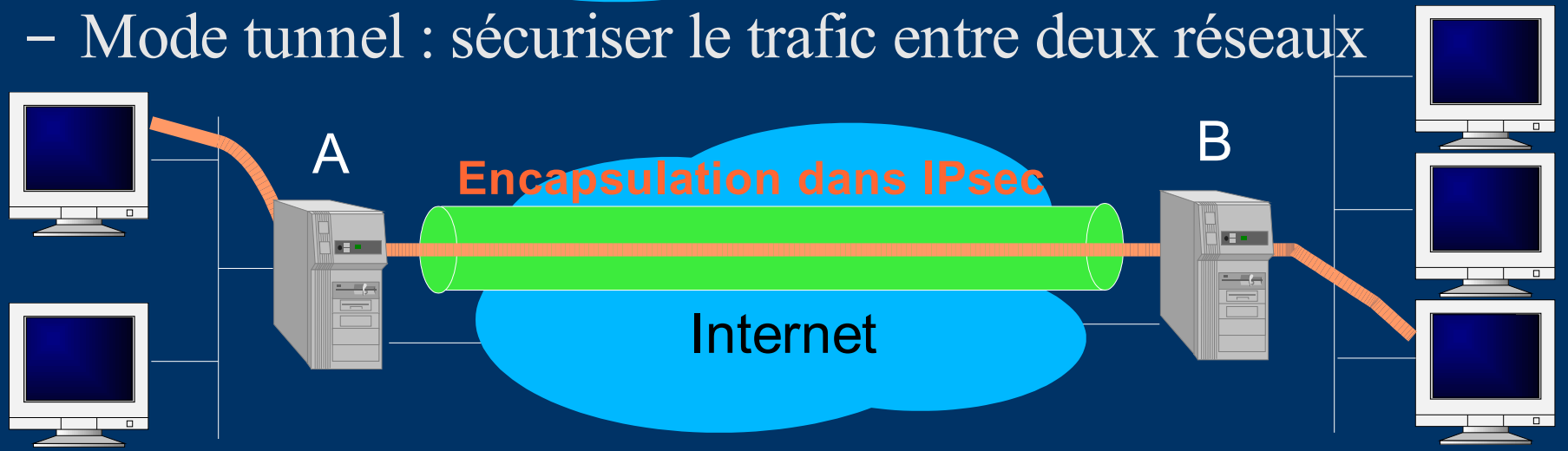
# Les tunnels IPsec (1)

- Objectifs d'IPsec
  - Issu d'IPv6 mais adapté à IPv4
  - Sécuriser le réseau de façon transparente pour l'utilisateur :
    - Authentification, chiffrement, intégrité sur l'ensemble des communisation
  - Négocier les protocoles de sécurité entre les deux parties
- Deux protocoles possibles :
  - AH (Authentication Header) : authentification, intégrité
  - ESP (Encapsulating Security Payload) : chiffrement

# Les tunnels IPsec (2)

- Deux modes possibles :
  - Mode transport : sécuriser le trafic entre deux machines


  - Mode tunnel : sécuriser le trafic entre deux réseaux



# Les tunnels IPsec (3)

- Phase 1 : association de sécurité ISAKMP
  - Utilise le port UDP 500
  - Négocie les algorithmes, authentifie les deux partenaires
  - Génére les clés pour les algorithmes (SHA1, DES3, DH)
  - Durée de vie limitée (quelques heures)
- Phase 2 : association de sécurité IPsec
  - Permet l'échange sécurisé des données
  - Echanges protégés par les clés générées dans la phase 1
  - Négocie les protocoles (AH ou ESP), compression, PFS
  - Durée de vie limitée (une heure)

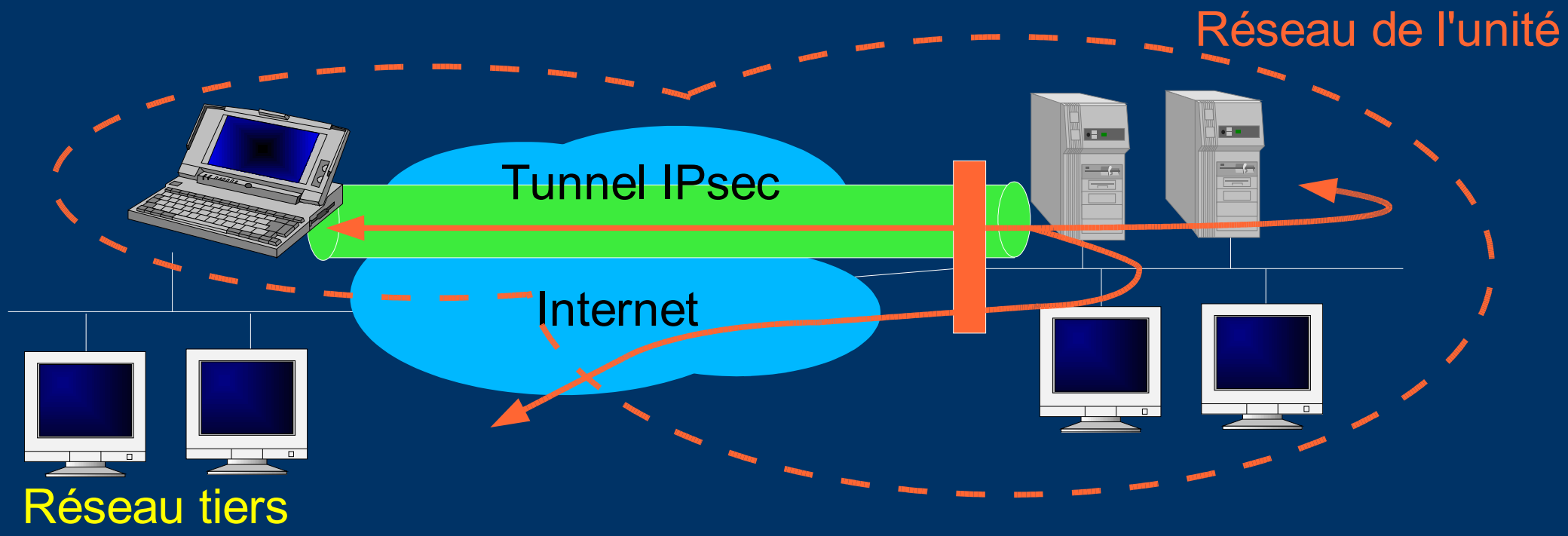
# *Les tunnels IPsec (4)*

Nombreuses implémentations :

- Pour Linux
  - En opensource : FreeSwan, OpenSwan, Racoon
  - Propriétaire : Cisco
- Pour Windows :
  - Standard à partir de Windows 2000
  - Propriétaire : IRE/Safenet, Cisco
- Pour MacOS X
  - En opensource : Racoon

# Les tunnels IPsec (5)

- Utilisation classique d'IPsec : le *road warrior*
  - Poste connecté dans un réseau tiers
  - Souhaite bénéficier de la protection du réseau de l'unité
  - Souhaite bénéficier d'une adresse dans ce réseau



# Les tunnels IPsec (6)

- Le modèle Microsoft / Cisco :
  - Objectif : sécuriser PPP. Passe par un empilement de couches :
    - IP sur PPP : modèle classique des connexions avec Windows
      - PPP authentifie l'utilisateur (login/mot de passe)
      - PPP alloue une adresse dans le réseau cible
      - PPP transporte les paquets IP
    - PPP sur L2TP
      - L2TP émule un lien de niveau 2 (RTC, LS, ...) sur IP
    - L2TP / Ipsec
      - IPsec authentifie les machines
      - IPsec sécurise tous les transferts

# Utilisation d'IPsec

- Le contexte de la Délégation
  - Peu d'utilisateurs (Délégué(e), ASR, ...)
  - Tous les portables en Windows 2000 ou XP
  - Aucun serveur Windows n'est autorisé à faire de l'IP avec l'extérieur
- Les solutions testées
  - 2000-2003 : client IRE/Safenet, serveur FreeSwan
    - Au début : authentification par clé privée partagée
    - Ensuite : authentification par certificat
  - 2004 : client Windows, serveur FreeSwan sur RedHat
  - 2005 : client Windows, serveur Racoon sur Fedora Core

# *Utilisation de Freeswan (1)*
## *sur Linux Redhat 9*

- Installation de FreeSwan sur RH 9
  - http://www.jacco2.dds.nl/networking/win2000xp-freeswan.html
  - Utilisation de RPMs :
    - Freeswan + patch X509
    - L2tpd
- Génération des certificats issus de l'IGC du CNRS
  - Un certificat machine pour le serveur FreeSwan
  - Un certificat machine pour chaque poste de travail
- Modification des fichiers de configuration pour
  - Ipsec (/etc/ipsec.conf, /etc/ipsec.secrets)
  - L2tpd et PPP (/etc/l2tpd/l2tpd2.conf, /etc/ppp/options.l2tpd)

# *Utilisation de Freeswan (2)*
## *sur Linux Redhat 9*

- Description de la passerelle IPsec
  - Un PC sous RedHat 9 : ipsec.domaine.fr
  - Adresse IP pour ISAKMP : 1.1.1.100, udp 500
  - L2TPd nécessite deux adresses :
    - Une externe : 1.1.1.100
    - Une interne : 1.1.1.101 (adresse IP virtuelle)
  - Les adresses IP dynamiques allouées aux postes distants :
    - Plage de 1.1.1.110 à 1.1.1.120

# Utilisation de Freeswan (3)
## Sur Linux Redhat 9 – Le fichier /etc/ipsec.conf

```
# basic configuration
config setup

# Disable Opportunistic Encryption
conn block
    auto=ignore

conn private
    auto=ignore

conn private-or-clear
    auto=ignore

conn clear-or-private
    auto=ignore

conn clear
    auto=ignore

conn packetdefault
    auto=ignore
```

# Utilisation de Freeswan (4)
## Sur Linux Redhat 9 – Le fichier /etc/ipsec.conf

```
# Connexion Windows 2000 ou XP, client IPsec mis a jour
conn portable-rd
  also=auth-by-cert
  right=%any
  rightrsasigkey=%cert
  rightid=/C=FR/O=Domaine/CN=client1.domaine.fr/emailAddress=xyz@domaine.fr
  leftprotoport=17/1701  # mettre 17/0 pour un IPsec non mis à jour
  rightprotoport=17/1701
  auto=add
  keyingtries=3

# Répéter autant de fois qu'il y a de connexions
# ...

conn auth-by-cert
  pfs=no
  authby=rsasig
  leftrsasigkey=%cert
  left=1.1.1.100
  leftcert=ipsec.domaine.fr.crt # relatif par rapport a /etc/ipsec.d/certs
  leftnexthop=%defaultroute
```

# Utilisation de Freeswan (5)
## Sur Linux Redhat 9 – Le répertoire /etc/ipsec.d

- Contient les certificats et clés privés
  - Répertoire `cacerts` :
    - Certificats des autorités de certification
  - Répertoire `certs` :
    - Certificat du serveur
  - Répertoire `crls` :
    - Liste de révocation des autorités de certification
  - Répertoire `private` :
    - Clé privée correspondant au certificat du serveur

# Utilisation de Freeswan (6)
## Sur Linux Redhat 9 – Le fichier /etc/ipsec.secrets

- Contient les clés secrètes, etc.
- Dans le cas d'une utilisation avec des certificats, indique le nom du fichier contenant la clé privée correspondant au certificat du serveur

```
: RSA ipsec.domaine.fr.key
```

# Utilisation de Freeswan (7)

## Sur Linux Redhat 9 – Fichier /etc/l2tpd/l2tpd.conf

```
[global]
listen-addr = 1.1.1.100

[lns default]
ip range = 1.1.1.110-1.1.1.120
local ip = 1.1.1.101
require chap = yes
refuse pap = yes
require authentication = yes
name = LinuxVPNserver
ppp debug = yes
pppoptfile = /etc/ppp/options.l2tpd
length bit = yes
```

# Utilisation de Freeswan (8)
## Sur Linux Redhat 9 – Fichier /etc/ppp/options.l2tpd

```
ipcp-accept-local
ipcp-accept-remote
ms-dns  1.1.1.1
ms-wins 1.1.1.10
domain  domaine.fr
auth
crtscts
idle 1800
mtu 1400
mru 1400
nodefaultroute
debug
lock
proxyarp
connect-delay 5000
ktune
```

Note: l'option `ktune` permet de configurer des paramètres du noyau, notamment le `ip_forward=1` obligatoire car la passerelle se comporte comme un routeur

# Utilisation de Freeswan (9)

## Sur Linux Redhat 9 – Fichier /etc/ppp/chap-secrets

```
# Secrets for authentication using CHAP
# client        server          secret                  IP addresses
user1           *               "le_mot_de_passe"       *
*               user1           "le_mot_de_passe"       *
user2           *               "le_mot_de_passe"       *
*               user2           "le_mot_de_passe"       *
user3           *               "le_mot_de_passe"       *
*               user3           "le_mot_de_passe"       *
```

# *Utilisation de Freeswan (10)*
## *Sur Linux Redhat 9*

- Contrôle des accès entrants vers la passerelle :
  - Ouverture du port UDP 500 vers la passerelle Ipsec
  - Ouverture des protocoles ESP(50) et AH(51)

- Penser à ouvrir ces accès sur les réseaux des invités

# Utilisation de Freeswan (11)
## Sur Linux Redhat 9

- Bilan
  - Solution délicate à mettre en oeuvre
    - Pour la partie serveur sous Linux
    - Pour la partie client sous Windows : utilisation de la console MMC pour configurer les certificats

    Heureusement il existe de bonnes documentations !
  - Compatible avec les certificats d'une IGC d'établissement
  - Malheureusement, Freeswan est en cours d'abandon =>
    - Tests de Racoon en cours
    - Tests de OpenSwan à venir

# *Utilisation de Racoon (1)*
## *Sur Fedora Core 2*

- Racoon est issu du projet KAME qui vise des implémentations de références d'IPv6 et d'IPsec pour IPv6 et Ipv4
- Racoon est l'implémentation d'ISAKMP/IKE
- Plateforme de développement : FreeBSD, OpenBSD, NetBSD, BSDI
- Utilise les couches basses d'IPsec fournies par l'OS => utilisable à partir de Linux 2.6 seulement

# *Utilisation de Racoon (2)*
## *Sur Linux Fedora Core 2*

- Au moment des tests, le RPM ipsec-tools est en version 0.2.5-4, mais la version stable est 0.3.3 =>
  - Installation à partir des sources
  - Création du fichier /etc/init.d/racoon
- Le fonctionnement avec des clients Windows est le même que précédemment
  - Installation de L2TPd à partir du même RPM
  - Reprise des mêmes fichiers de configuration

# Utilisation de Racoon (3)
## Sur Linux Fedora Core 2

- Configuration de racoon
  - Fichier /etc/racoon/racoon.conf
    - Voir planches suivantes
  - Répertoire /etc/racoon/certs contenant obligatoirement :
    - Les certificats de l'autorité de certification
    - Les listes de révocation (il faut les maintenir à jour !)
    - La clé privée du serveur
    - Le certificat du serveur

# Utilisation de Racoon (4)
## Sur Linux Fedora Core 2 – Fichier racoon.conf

```
path include "/etc/racoon";
path certificate "/etc/racoon/certs";

listen {
    isakmp 1.1.1.100 [500];
    strict_address;
}

remote anonymous {
    exchange_mode main;
    doi ipsec_doi;
    situation identity_only;
    generate_policy on;
    lifetime time 28800 sec;

    my_identifier asn1dn;
    certificate_type x509 "ipsec.domaine.fr.crt" "ipsec.domaine.fr.key";
    peers_identifier asn1dn;
    verify_cert on;

    proposal {
        encryption_algorithm 3des;
        hash_algorithm sha1;
        authentication_method rsasig;
        dh_group modp1024;
    }
}
```

# Utilisation de Racoon (5)
## Sur Linux Fedora Core 2 – Fichier racoon.conf

```
sainfo anonymous from asn1dn
   "C=FR,O=Domaine,CN=client1.domaine.fr,emailAddress=xyz@domaine.fr"
   {
    encryption_algorithm 3des, blowfish 448, rijndael ;
        lifetime time 28800 sec;
        encryption_algorithm 3des ;
        authentication_algorithm hmac_md5, hmac_sha1 ;
        compression_algorithm deflate ;
}


# répeter autant de fois qu'il y d'utilisateurs
# ...
```

# Utilisation de racoon (6)
## Sur Linux Fedora Core 3

- Problèmes rencontrés avec Racoon
  - Déconnexion après renégociation en fin de vie des SA
  - Reconnaissance de la chaine de certification :
    - Lors de l'authentification des partenaires, Racoon s'eattend à recevoir le certificat du client, et lui seul
    - Windows utilise l'algorithme suivant :
      - Si le certificat du client est signé par une autorité racine, Windows ne transmet que le certificat du client
      - Si le certificat du client est signé par une autorité intermédiaire, Windows envoie la chaine de certification = le certificat du client + le certificat de toutes les autorités intermédiaires
    - L'IGC du CNRS délivre des certificats qui sont dans le 2e cas
      - Création d'un patch (pas très difficile)

# Bilan de l'utilisation de IPsec

- Solution complexe à mettre en oeuvre :
  - Interopérabilité hasardeuse :
    - Une solution pour Windows, une pour Linux, une pour ... => une configuration spécifique de la passerelle IPsec pour chacune d'elles
  - Attrait des solutions commerciales client+serveur :
    - Exemple : client IPsec + PIX Cisco
- Contrôles d'accès trop rigoureux sur le réseau hôte
  - UDP 500 est en général fermé
  - ESP et AH ont été oubliés dans les configs de Firewall
- Non support de NAT => un autre procotole !

# *Les tunnels SSL (1)*

- Au départ, SSL (*Secure Socket Layer*) est destiné à sécuriser les connexions TCP entre un client et un serveur. Exemples :
  - HTTP / SSL entre un navigateur et un serveur WWW
  - IMAP / SSL entre un client et un serveur de messagerie
  - SMTP / TLS entre deux serveurs de messagerie

- Stunnel (`www.stunnel.org`) :
  - Permet d'utiliser SSL lorsque l'application sur le client et/ou le serveur ne sait pas faire du SSL
    - Sécuriser des accès à des bases de données
    - Sécuriser rsync, LDAP, VNC, ...

# *Les tunnels SSL (2)*

- Autre idée : utiliser SSL en lieu et place d'IPsec pour faire des VPN

- Exemple : OpenVPN (www.openvpn.org)
  - Produit OpenSource, licence GPL
  - Utilise les interfaces virtuelles TUN/TAP
  - Encapsule le trafic dans une connexion SSL
  - Utilise UDP (port 1194) plutôt que TCP
    - Problèmes liés à TCP dans TCP

# *Utilisation de OpenVPN (1)*
## *Sur Linux Fedora Core 2*

- Installation à partir de la distribution source
  - Contient les instructions pour fabriquer un RPM
  - Installation du RPM
  - Configuration : répertoire /etc/openvpn
    - Installation des certificats des autorités de certification
    - Installation du certificat du serveur
    - Installation de la clé privée correspondante
    - Génération des paramètres d'échange de clés (Diffie-Hellman)
    - Adaptation du script server.conf

# *Utilisation de OpenVPN (2)*
## *Sur Linux Fedora Core 2*

- Deux modes de fonctionnement :
  - Mode tunnel : niveau 3 (routage)
    - Le client obtient une adresse IP de la part du serveur OpenVPN. Cette adresse est dans un réseau privé.
    - Le trafic entre le client, le serveur, le réseau de l'unité est routé
    - Les broadcasts ne sont pas acheminés vers le tunnel
  - Mode pont : niveau 2
    - Le client distant est vu comme étant directement connecté au réseau local de l'unité
    - Obtient une adresse IP de la même façon que les postes directement connectés au réseau local (DHCP, statique)
    - Les broadcasts sont acheminés dans le tunnel

# Utilisation de OpenVPN (3)

## Sur Linux Fedora Core 2 – Le fichier server.conf

```
# configuration pour OpenVPN en mode tunnel
local 1.1.1.100
proto udp
dev tun
ca ca.crt
cert ipsec.domaine.fr.crt
key ipsec.domaine.fr.key  # This file should be kept secret
dh dh1024.pem # fichier obtenu par : openssl dhparam -out dh1024.pem 1024
server 10.15.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway"
push "dhcp-option DNS 1.1.1.1"
push "dhcp-option WINS 1.1.1.10"
keepalive 10 120
tls-verify "/etc/openvpn/verify-dn"
comp-lzo
max-clients 10
user nobody
group nobody
persist-key
persist-tun
status openvpn-status.log
verb 3
```

# *Utilisation de OpenVPN (4)*
## *Sur Linux Fedora Core 2 – Le script verify-dn*

- Optionnel
- Permet d'autoriser ou non la connexion en fonction du titulaire du certificat
  - Interface: `verify-dn` *profondeur sujet_du_certificat*
  - Par exemple :
    - Vérifier que le sujet du certificat commence par : /C=FR/O=CNRS/OU=*mon_labo*
      afin de n'autoriser que les utilisateur de *mon_labo* à se connecter au VPN

# Utilisation de OpenVPN (5)
## Sur Linux Fedora Core 2 – Contrôle d'accès

- OpenVPN encapsule le trafic dans UDP
  - Côté serveur : port UDP 1194
  - Côté client :
    - Port UDP fixé dans la configuration
    - Port UDP dynamique

- Contrôles d'accès :
  - Côté serveur (donc, sur le réseau que vous gérez)
    - Autoriser UDP 1194 vers le serveur OpenVPN
  - Côté client (donc, sur un réseau que vous ne gérez pas !)
    - Problème si le client utilise des ports dynamiques
    - Solution : utiliser un port fixe, exemple 1194 sur le client

# Utilisation de OpenVPN (6)
## Sur Linux Fedora Core 2 – Routage

- Dans une configuration en mode tunnel
  - Le poste distant obtient une adresse privée (10.x.y.z/24)
  - Il se présente avec cette adresse sur les machines internes
    - Problème de routage : obligation de mettre une route vers 10.x.y.0/24 sur toutes les machines
    - Problème avec la reconnaissance sur adresse IP (tcp_wrapper)
  - Solution

    Sur Linux, utiliser le *masquerading* de iptables :

```
iptables -t nat -A POSTROUTING \
    -s 10.15.0.0/255.255.255.0 \
    -j SNAT -to-source 1.1.1.101
```

- Trop compliqué ?
  - Utiliser le mode pont
  - Attention aux connexions sur des réseaux bas-débit

# *Utilisation de OpenVPN (7)*
## *Sur Linux Fedora Core 2 – Le client*

- Configuration d'un client sous Windows 2000
  - Installation de OpenVPN + client graphique : http://openvpn.se/download.html
  - Répertoire `C:\Program Files\OpenVPN\config`
    - Fichier `client.ovpn`
    - Copie des autorités de certification dans `ca.crt`
    - Sauvegarde du certificat dans `Prenom_ Nom.p12`

# Utilisation de OpenVPN (8)
## Sur Linux Fedora Core 2 – Fichier client.ovpn

```
# configuration d'un client en mode tunnel
client
dev tun
proto udp
remote ipsec.domaine.fr 1194
lport 1194
resolv-retry infinite
persist-key
persist-tun
ca ca.crt
pkcs12 Prenom_Nom.p12
ns-cert-type server
comp-lzo
verb 3
```

# Utilisation de OpenVPN (9)
## Sur Linux Fedora Core 2

- Bilan de l'utilisation de OpenVPN
  - Très simple à configurer
    - Même fichier de configuration pour client et serveur
    - Repose sur des technologies bien connues : SSL, certificats, ...
  - Problèmes rencontrés avec le client
    - Fonctionne avec des certificats CNRS d'utilisateur, mais pas avec des certificats de machines
    - Fonctionnement du pare-feu Windows XP incompréhensible
    - Sous Windows, il faut être administrateur de la machine pour démarrer le tunnel :
      - Possibilité d'utiliser un service, mais la clé privée doit être stockée en clair

# Conclusion

- Un éventail de solutions robustes aux ergonomies accessibles à tout utilisateur
- L'environnement du réseau hôte peut empêcher une solution de fonctionner :
  - Blocage du port 25 en sortie : ~~SMTP/TLS~~
  - Blocage du trafic UDP 500, ESP, AH : ~~IPsec~~
  - Adresses privées + NAT : ~~IPsec~~
  - Blocage de UDP 1194 : ~~OpenVPN~~
  - Adresses privées + proxy HTTP : ~~SSH~~, ~~IMAP/SSL~~
- Une solution cible (ex. IPsec ou OpenVPN), dégradation progressive en SSH, clients SSL, ..., webmail