Virtual Private Networks

Network Infrastructures labs

Marco Spaziani Brunella

SAPIENZA
UNIVERSITÀ DI ROMA

Lecture details

- **Readings:**
  - Virtual Private Networks; Charlie Scott et al.; O'Reilly
- **Lecture outline:**
  - Private networks
  - Virtual private networks

Private Network

"A private network is composed of computers owned by a single organization that share information specifically with each other. They're assured that they are going to be the only ones using the network, and that information sent between them will (at worst) only be seen by others in the group. The typical corporate Local Area Network (LAN) or Wide Area Network (WAN) is an example of a private network. The line between a private and public network has always been drawn at the gateway router, where a company will erect a firewall to keep intruders from the public network out of their private network, or to keep their own internal users from perusing the public network."

Private Network issues

Suppose you're working for a corporate and you have all your personal files stored in your PC at work that send you on a mission abroad.
How can you get your sensible files while you are on your laptop abroad?
Two options may arise:

- Use a dedicated P2P connection → too costly
- Use the public internet → Security of data? Managing of private IPs?

Virtual Private Network

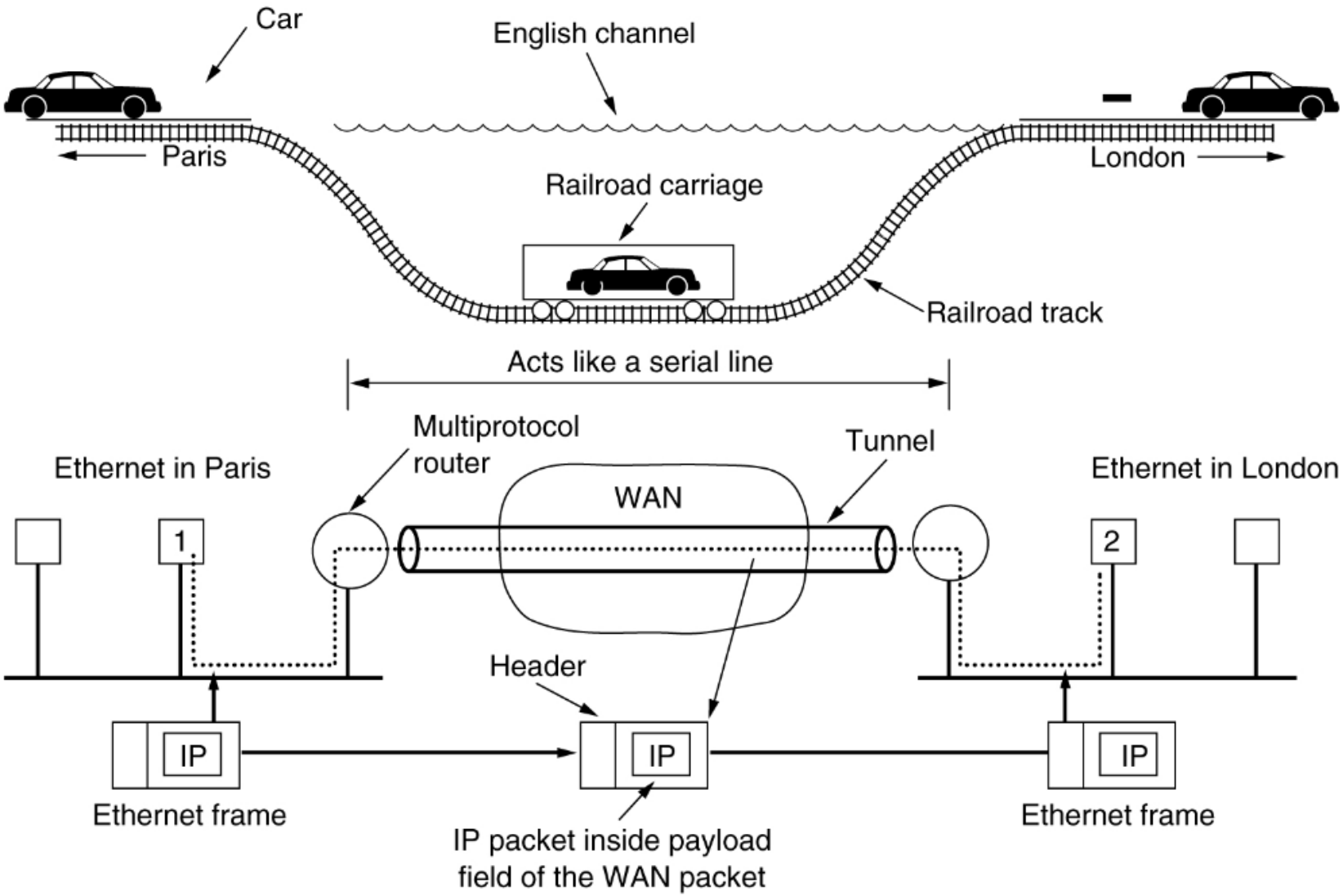
"VPNs allow you to create a secure, private network over a public network such as the Internet. They can be created using software, hardware, or a combination of the two that creates a secure link between peers over a public network. This is done through encryption, authentication, packet tunneling and firewalls."

"Since the Internet is a public network, you always risk having someone access any system you connect to it."

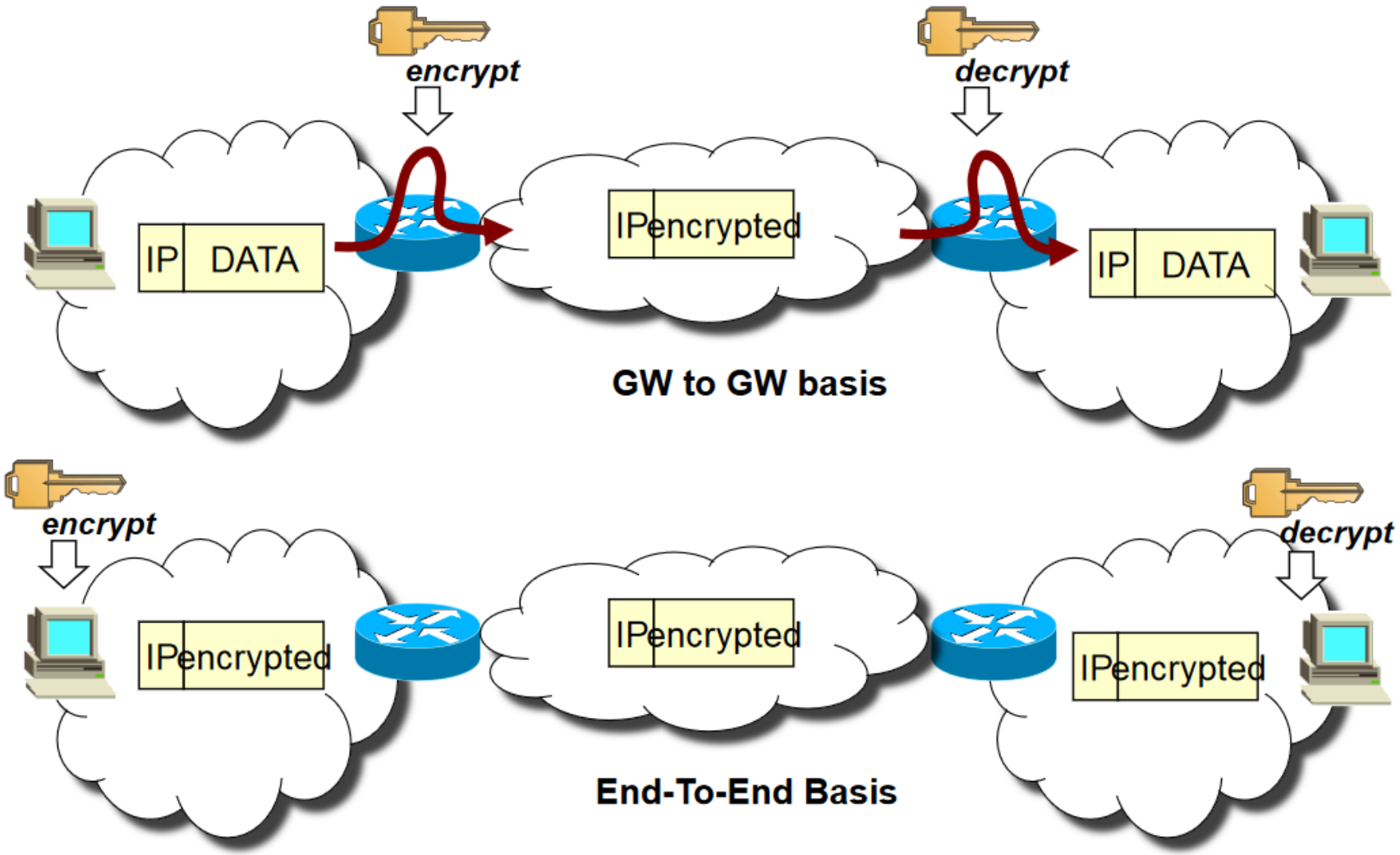
Also VPNs have issues, in fact:

"Running a virtual private network over the Internet raises an easily forgotten issue of reliability. Let's face it: the Internet isn't always the most reliable network, by nature. Tracing a packet from one point to another, you may pass through a half-dozen different networks of varying speeds, reliability, and utilization—each run by a different company. Any one of these networks could cause problems for a VPN."

Tunnels


Car
English channel
Paris
London
Railroad carriage
Railroad track
Acts like a serial line
Multiprotocol router
Tunnel
Ethernet in Paris
Ethernet in London
WAN
1
2
Header
IP
IP
IP
Ethernet frame
Ethernet frame
IP packet inside payload field of the WAN packet

VPN Modes


encrypt
decrypt
IP DATA
IPencrypted
IP DATA
GW to GW basis
encrypt
decrypt
IPencrypted
IPencrypted
IPencrypted
End-To-End Basis

OpenVPN

We will use OpenVPN as VPN implementation on Linux.
Some features are:

- tunnel any IP subnetwork or virtual ethernet adapter over a single UDP or TCP port
- use any cipher, key size, or HMAC digest (for datagram integrity checking) supported by the OpenSSL library
- choose between static-key based conventional encryption or certificate-based public key encryption
- tunnel networks over NAT
- use static, pre-shared keys or TLS-based dynamic key exchange

Public Key Certificate

A public key certificate is a data structure that binds a public key (and therefore the related private key) to the the identity of the legitimate owner:

$$CERT_{id} = \langle id, PUB\_KEY_{id} \rangle$$

The binding between $\langle id, PUB\_KEY_{id} \rangle$ is granted by a trusted Certification Authority that signs $CERT_{id}$.

Provided that we have the CA's public key, we can verify the CA signature and therefore verify the public key authenticity

Public Key Infrastructure

A PKI consists of the protocols, the policies and the cryptography mechanism used to manage public key certificate.

A PKI requires the definition of:

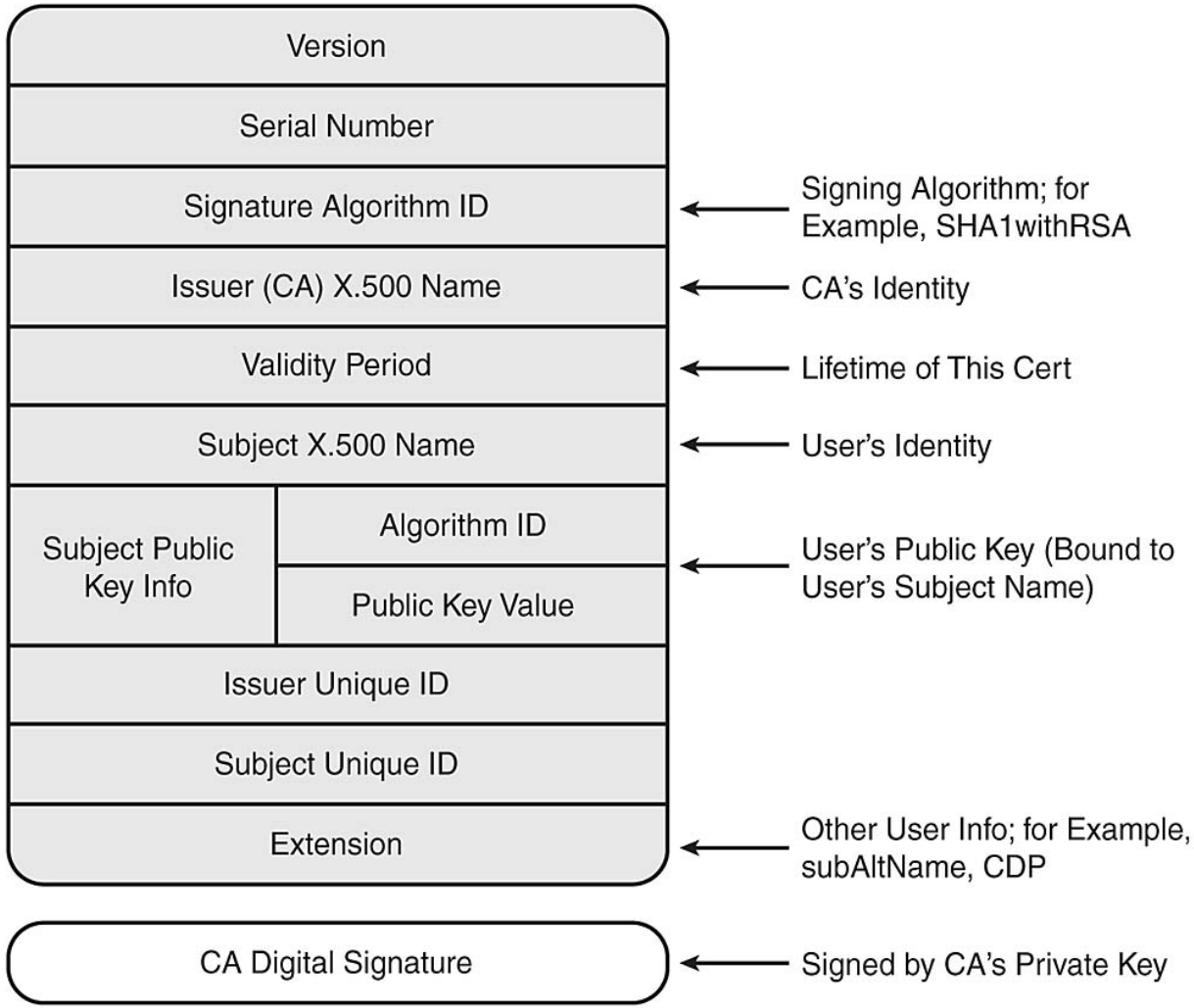
- Certificate format
- Relationship among CAs
- Mechanisms and policies for issuing and revoking certificate
- Storage services

Who issues the certificate of a CA? Another CA !!!

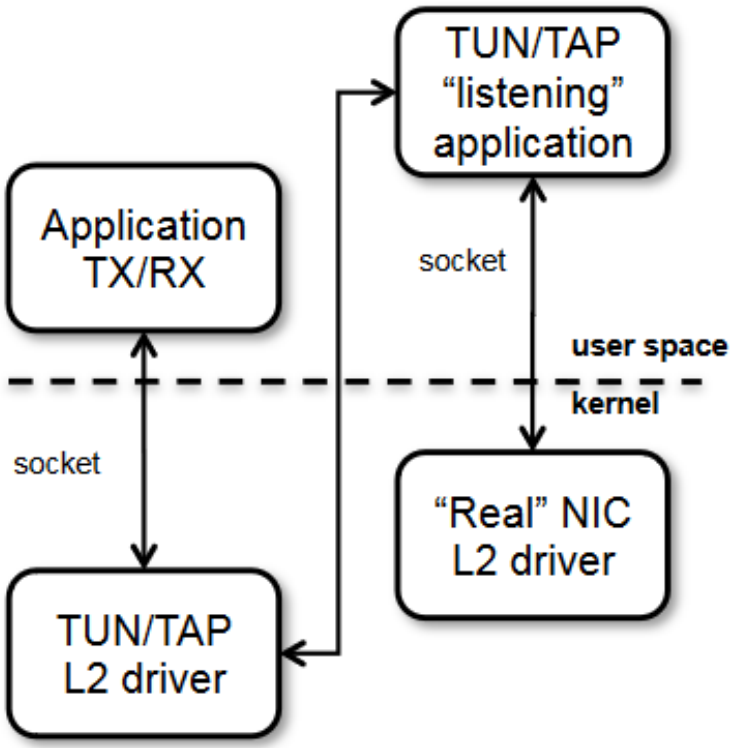
"All animals are equal, but some animals are more equal than others"

Key concept: You have to trust CAs!

X.509 certificate format

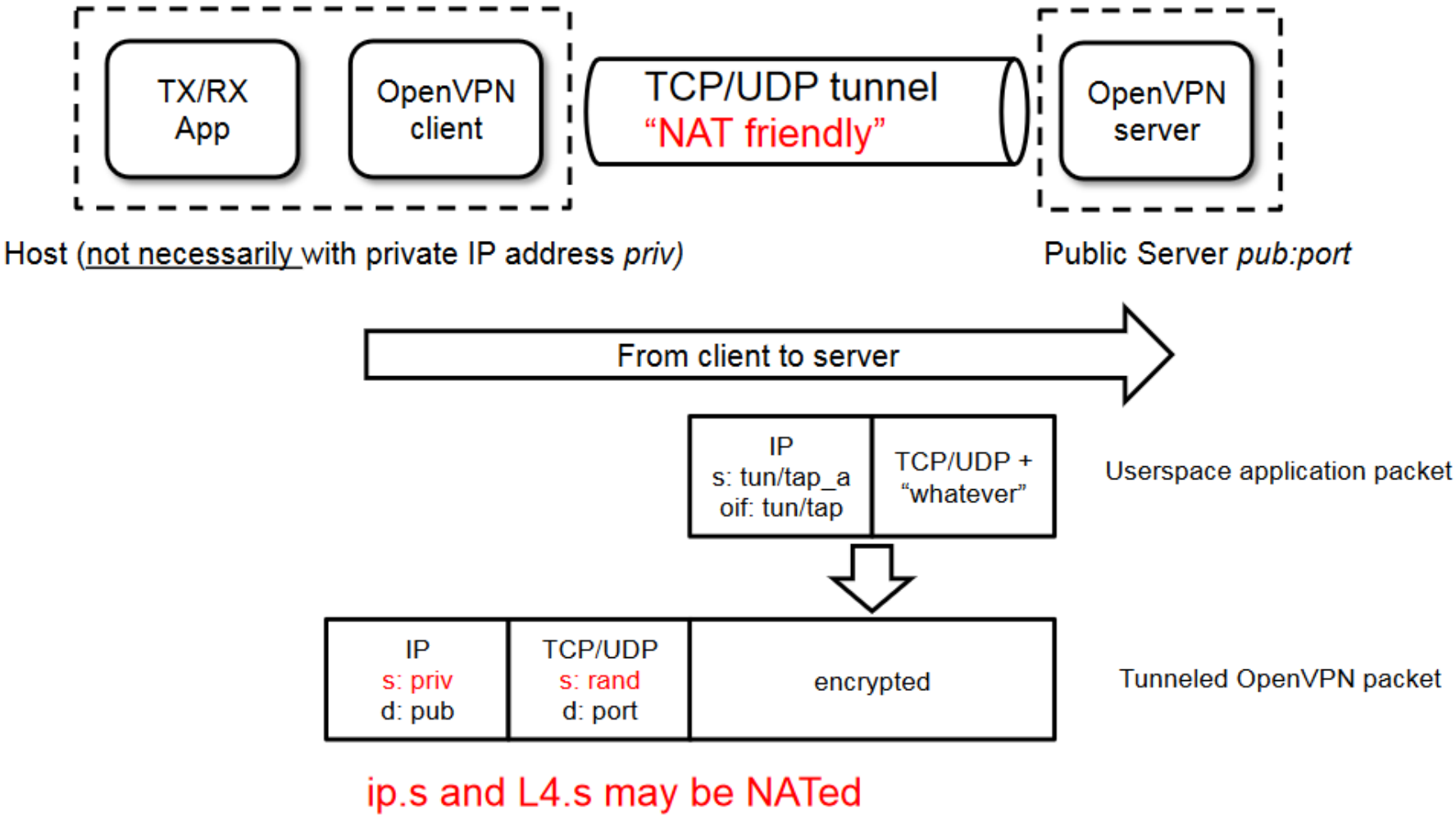


TUN/TAP Drivers



- TUN is a virtual Point-to-Point network device for IP tunneling
- TAP is a virtual Ethernet network device for Ethernet tunneling
- Userland application can write {IP|Ethernet} frame to /dev/{tun|tap}X and kernel will receive this frame from {tun|tap}X interface
- In the same time every frame that kernel writes to {tun|tap}X interface can be read by userland application from {tun|tap}X device

OpenVPN Architecture



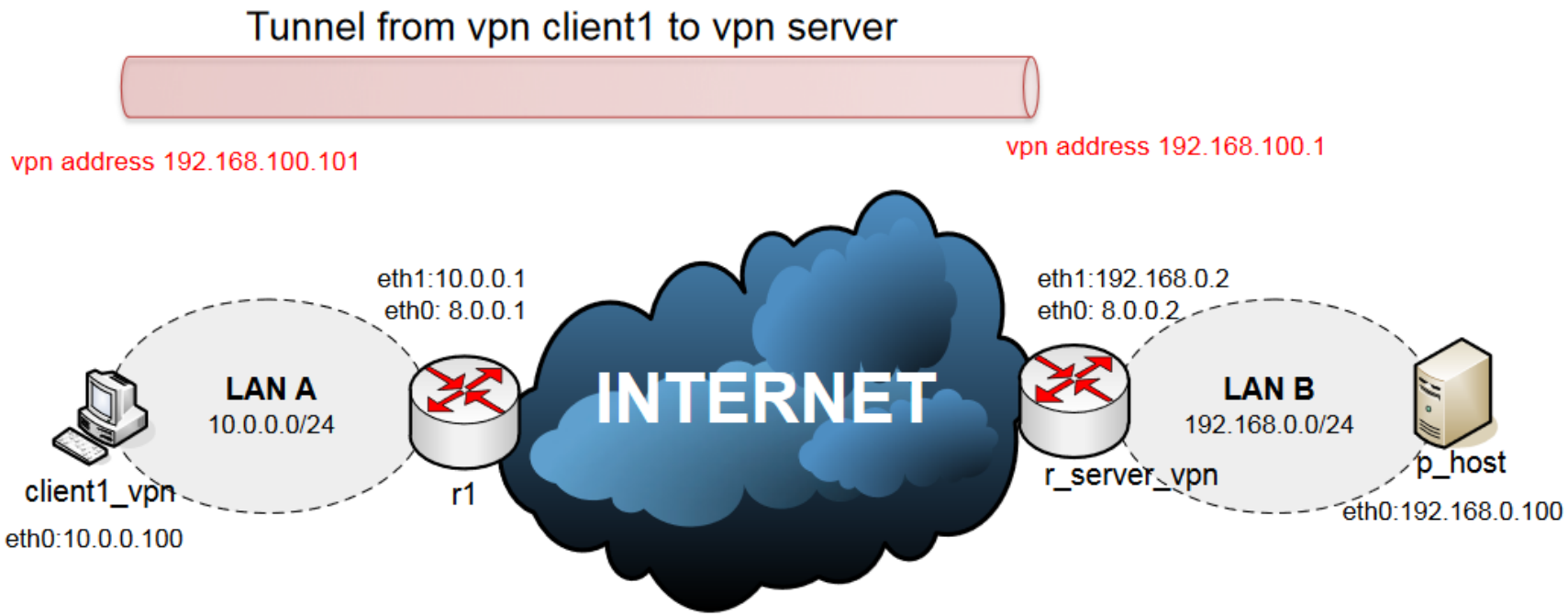
OpenVPN PKI

- The first step in building an OpenVPN 2.0 configuration is to establish a PKI which consists of:
  - a separate certificate (also known as a public key) and private key for the server and each client
  - a master Certificate Authority (CA) certificate and key which is used to sign each of the server and client certificates
- OpenVPN supports bidirectional authentication based on certificates, meaning that the client must authenticate the server certificate and the server must authenticate the client certificate before mutual trust is established
- Both server and client will authenticate the other by first verifying that the presented certificate was signed by the master certificate authority (CA), and then by testing information in the now-authenticated certificate header, such as the certificate common name or certificate type (client or server)

OpenVPN Security model

- This security model has a number of desirable features from the VPN perspective:
  - The server only needs its own certificate/key -- it doesn't need to know the individual certificates of every client which might possibly connect to it.
  - The server will only accept clients whose certificates were signed by the master CA certificate (which we will generate below). And because the server can perform this signature verification without needing access to the CA private key itself, it is possible for the CA key (the most sensitive key in the entire PKI) to reside on a completely different machine, even one without a network connection.
  - If a private key is compromised, it can be disabled by adding its certificate to a CRL (certificate revocation list). The CRL allows compromised certificates to be selectively rejected without requiring that the entire PKI be rebuilt.
  - The server can enforce client-specific access rights based on embedded certificate fields, such as the Common Name.

lab_vpn_1



VPN network: 192.168.100.0/24

Building the Certification Authority

- We'll use set of scripts bundled with OpenVPN
  - In netkit, cd into the directory: `/usr/share/doc/openvpn/example/easy-rsa/2.0`
- Edit the `vars` file on r_server_vpn

Initialize the PKI

```
r_server_vpn# . ./vars
r_server_vpn# ./clean-all
r_server_vpn# ./build-ca
```

The final command (build-ca) will build the certificate authority (CA) certificate and key by invoking the interactive openssl command. Most queried parameters were defaulted to the values set in the vars file. The only parameter which must be explicitly entered is the Common Name.

Server and client

Generate a certificate and private key for the server

```
r_server_vpn# ./build-key-server server
```

Generate client keys and certificates

```
r_server_vpn# ./build-key client1
```

Diffie Hellman parameters must be generated for the OpenVPN server

```
r_server_vpn# ./build-dh
```

Meaning of files generated

Filename	Needed By	Purpose	Secret
ca.crt	server + all clients	Root CA certificate	NO
ca.key	key signing machine only	Root CA key	YES
dh{n}.pem	server only	Diffie Hellman parameters	NO
server.crt	server only	Server Certificate	NO
server.key	server only	Server Key	YES
client1.crt	client1 only	Client1 Certificate	NO
client1.key	client1 only	Client1 Key	YES

Server Configuration

```
port 1194
proto udp
dev tun
cert /root/server.crt
key /root/server.key
dh /root/dh.pem
server 192.168.100.0 255.255.255.0
push "route 192.168.0.0 255.255.255.0"
client-config-dir /root/ccd
client-to-client
keepalive 10 120
comp-lzo
persist-key
persist-tun
status openvpn-status.log
verb 3
```

```
r_server_vpn~# openvpn server.conf
```

Client Configuration Directory

- A file for each OpenVPN client “CN”
  - In this lab: client1
- In each file (+ other commands we’re not considering):
  - if-config-push [local_ptp] [remote_ptp]
  - iroute [net_addr] [net_mask]
- client1
  - `if-config-push 192.168.100.101 192.168.100.102`

Allowed /30 pairs

```
[  1,  2] [  5,  6] [  9, 10] [ 13, 14] [ 17, 18]
[ 21, 22] [ 25, 26] [ 29, 30] [ 33, 34] [ 37, 38]
[ 41, 42] [ 45, 46] [ 49, 50] [ 53, 54] [ 57, 58]
[ 61, 62] [ 65, 66] [ 69, 70] [ 73, 74] [ 77, 78]
[ 81, 82] [ 85, 86] [ 89, 90] [ 93, 94] [ 97, 98]
[101,102] [105,106] [109,110] [113,114] [117,118]
[121,122] [125,126] [129,130] [133,134] [137,138]
[141,142] [145,146] [149,150] [153,154] [157,158]
[161,162] [165,166] [169,170] [173,174] [177,178]
[181,182] [185,186] [189,190] [193,194] [197,198]
[201,202] [205,206] [209,210] [213,214] [217,218]
[221,222] [225,226] [229,230] [233,234] [237,238]
[241,242] [245,246] [249,250] [253,254]
```

Client Configuration

```
client
dev tun
proto udp
remote 8.0.0.2 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca /root/ca.crt
cert /root/client1.crt
key /root/client1.key
ns-cert-type server
comp-lzo
verb 3
```

```
client1_vpn~# openvpn client.conf
```

Why push route?

- Without that a packet from client1_vpn to 192.168.0.0/24 would be delivered to r1
- r1 would discard it as it would not know the route
- With push route client1_vpn packets to 192.168.0.0/24 are correctly send out through tun0 and properly delivered via the OpenVPN tunnel