

# OpenVPN

國立陽明交通大學資工系資訊中心

Computer Center of Department of Computer Science, NYCU

# Why OpenVPN

1. Cross-platform portability
  - a. iOS / Android / Windows / Linux / FreeBSD
  - b. OpenWRT
2. Extensible VPN framework
  - a. Logging
  - b. Authentication
3. OpenVPN uses an [industrial-strength security model](#)

# TUN/TAP

- TAP

- Layer 2
- behave like adapter
- More overhead(L2)
- Transfer any protocol
- Bridge

- TUN

- Layer 3
- Less Overhead(L3)
- Only IPv4 , IPv6(OpenVPN 2.3)
- No Bridges!

# Configuring OpenVPN

- A server/client setting can be described as a `ovpn/conf` file.
- At most circumstances, we will separate `key/ca` files to make config file clean.

# Configuration

- `/usr/local/etc/openvpn/openvpn.conf`
  - copy
    - From: `/usr/local/share/examples/openvpn/sample-config-files/server.conf`
    - To: `/usr/local/etc/openvpn/openvpn.conf`
- In `/etc/rc.conf.local`
  - `openvpn_enable="YES"`
  - `openvpn_configfile="/usr/local/etc/openvpn/openvpn.conf"`

# A simple server config(1/2)

```
port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key # This file should be kept secret
dh dh2048.pem
topology subnet
server 192.168.14.0 255.255.255.0
ifconfig-pool-persist ipp.txt
client-config-dir static_clients
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"
client-to-client
```

# A simple server config(2/2)

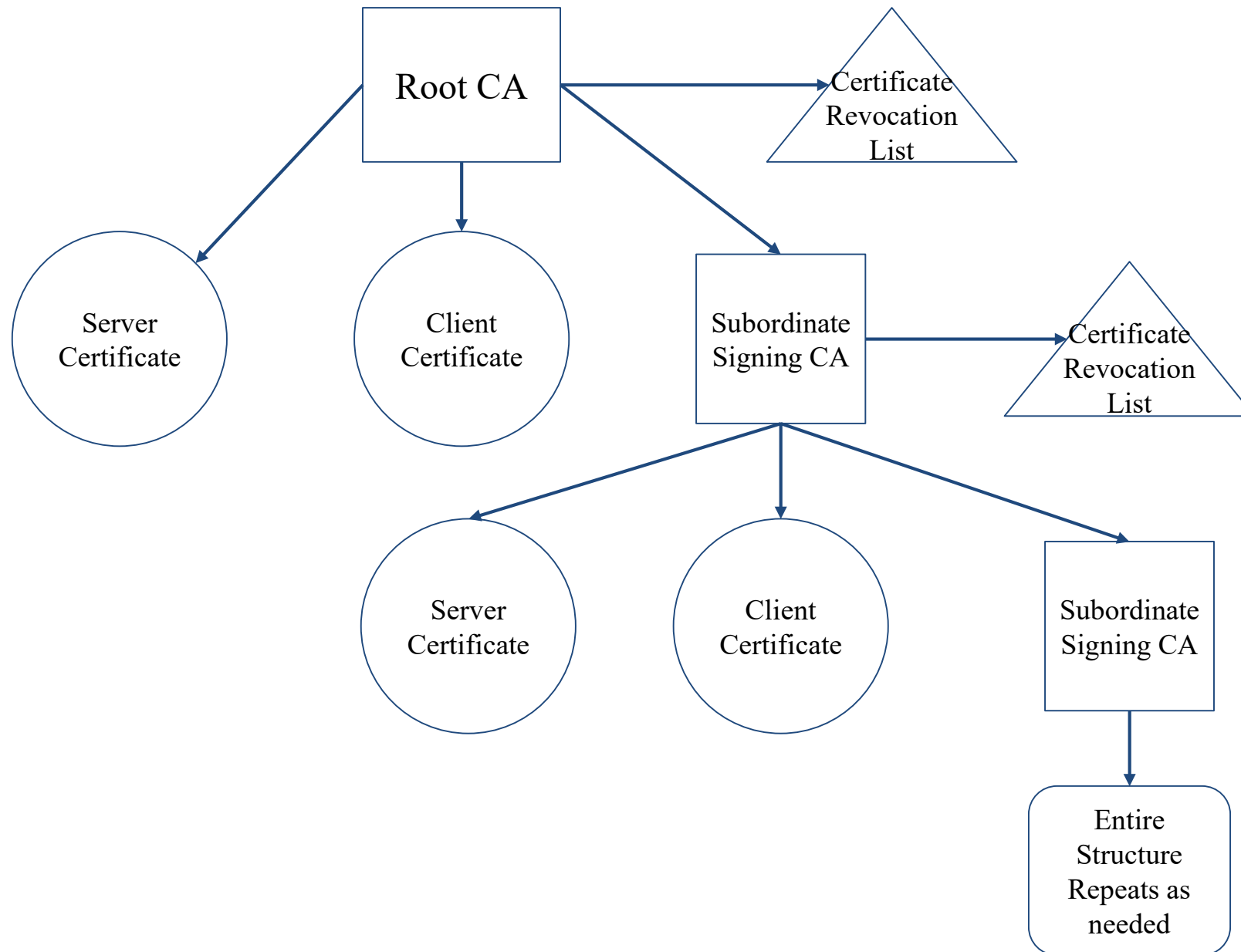
```
keepalive 10 120
tls-auth ta.key 0 # This file is secret
cipher AES-256-CBC # AES
comp-lzo
max-clients 10
user nobody
group nobody
persist-key
persist-tun
verb 5
mute 20
```

# A simple client config

```
client
dev tun
proto udp
remote xxx.com 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client.crt
key client.key
remote-cert-tls server
tls-auth ta.key 1
cipher AES-256-CBC
comp-lzo
verb 3
mute 20
```



# X.509 PKI



# Diffie Hellman parameters

- Diffie–Hellman is used to secure a variety of [Internet](#) services. However, research published in October 2015 suggests that the parameters in use for many D-H Internet applications at that time are not strong enough to prevent compromise by very well-funded attackers, such as the security services of large governments. ([wikipedia](#))
- Generate 2048-bit dhparams!

# HMAC

- tls-auth
- The tls-auth directive adds an additional HMAC signature to all SSL/TLS handshake packets for integrity verification. Any UDP packet not bearing the correct HMAC signature can be dropped without further processing. The tls-auth HMAC signature provides an additional level of security above and beyond that provided by SSL/TLS. It can protect against:
  - DoS attacks or port flooding on the OpenVPN UDP port.
  - Port scanning to determine which server UDP ports are in a listening state.
  - Buffer overflow vulnerabilities in the SSL/TLS implementation.
  - SSL/TLS handshake initiations from unauthorized machines (while such handshakes would ultimately fail to authenticate, tls-auth can cut them off at a much earlier point).

# Generate ca, cert

1. Use easy-rsa, an openvpn ca,cert generate tool
2. Do it from scratch with openssl

※ Question: Can we generate certificates using Let's Encrypt? Pros & Cons ?

# easy-rsa

- In FreeBSD:

```
# pkg install easy-rsa

# mkdir /root/ca
# cd /root/ca
# easyrsa init-pki
# easyrsa build-ca

# cd /usr/local/etc/openvpn/
# easyrsa init-pki
# easyrsa gen-req [NAME] nopass
# easyrsa gen-dh

# mkdir /root/client
# cd /root/client
# easyrsa init-pki
# /easyrsa gen-req [NAME]
```

# Sign key to CA

```
# cd /root/ca
# easyrsa import-req /usr/local/etc/openvpn/pki/reqs/[NAME].req [NAME]
# easyrsa import-req /root/client/pki/reqs/[NAME].req [NAME]

# easyrsa sign-req server [NAME]
# easyrsa sign-req client [NAME]
```

# Diffie-Hellman / TLS-auth key

## DH-KEY

```
# cd /usr/local/etc/openvpn  
# easyrsa gen dh
```

## AUTH KEY (Server & Client)

```
# cd /usr/local/etc/openvpn  
# openvpn -genkey -secret ta.key
```

# Package your config

- Server

- ca.crt
- server.conf
- server.key
- server.crt
- dh.pem
- ta.key

- Client

- ca.crt
- client.conf
- client.key
- client.crt
- ta.key



# Enable and start

- SERVER SIDE

```
# cp keys,conf,crts... /usr/local/etc/openvpn  
# /usr/local/etc/rc.d/openvpn start
```

- CLIENT SIDE

```
# cp keys,conf,crts... /usr/local/etc/openvpn  
# /usr/local/etc/rc.d/openvpn start
```

# User-authentication

1. Simply by signing client certs.
2. Use Username/password
3. Use 3rd party authentication
  - RADIUS
  - LDAP

# Server Side

```
Inside server.conf
```

```
# Using PAM to auth (Working with LDAP/NIS/Local Account)
(verify-client-cert)
```

```
plugin /usr/local/lib/openvpn/plugins/openvpn-plugin-auth-pam.so login
```

```
# Use a shell script to auth
```

```
auth-user-pass-verify /etc/openvpn/auth.sh via-env
```

```
script-security 3 # To allow script reading passwords
```

Reference:

- /usr/share/doc/openvpn-2.4.6/README.auth-pam
- /etc/pam.d/login

# Client Side

```
# A dialog will popup to ask you username/password
auth-user-pass
# Saving username/password into a file
auth-user-pass client.secret
# cat client.secret
ClientName
ClientPassword
```

# Reference

- <https://www.digitalocean.com/community/tutorials/how-to-setup-and-configure-an-openvpn-server-on-centos-7>
- <https://www.howtoforge.com/tutorial/how-to-install-openvpn-on-centos-7/>
- <https://wiki.archlinux.org/index.php/OpenVPN>