# HostAP WPA Workshop

Jan Fiegert

12. Dezember 2004

## 1 WPA, 802.1x und EAP

802.11 basierte Verfahren sind seit vielen Jahren der Standard zum Betrieb drahtloser Funknetze. Die dabei zum Einsatz kommenden Verschlüsselungstechnik WEP erwies sich allerdings recht bald als unbrauchbar für Szenarios in denen echte Sicherheit auf dem Übertragungsmedium erforderlich ist. Dabei ist weniger der Algorithmus (RC4) selbst das Problem. Die eigentliche Schwäche liegt in der Art und Weise wie er bei WEP zum Einsatz kommt.

WPA ist angetreten die Implementierungsschwächen in WEP auszumerzen ohne einen völlig neuen Algorithmus zu verwenden. Aufgrund dessen kann die verwendete Hard- und Firmware weiter verwendet werden. Ein neuer Verschlüsselungsalgorithmus ist für die nächste Version des 802.11 Standards (802.11i) vorgesehen.

Um die Schwachstellen von WEP in der zwischen Zeit auszumerzen, hat die WI-FI Allianz das Verfahren *Wi-Fi Protected Access*(kurz WPA) definiert. WPA kombiniert dazu neue Authentifizierungs Verfahren mit bereits implementierten Verschlüsselungsalgorithmen. Der 802.1x Standard beschreibt die folgenden Teilnehmer:

- **Supplicant:** Hierbei handelt es sich um die 802.1x Client Komponente. Er implementiert Schlüsselaustausch und Authentifizierung. Supplicants existieren für alle gängigen Betriebssysteme. (xsupplicant und Open1x für Linux, eapol für Mac OS X, die neueren Windows Systeme enthalten ebenfalls einen Supplicant für 802.1x)

- **Authenticator:** Dieses Komponente regelt den Zugriff auf das zu schützenden Netzwerk Segment. Solange eine Client nicht authentifiziert ist, gestattet er nur Traffic, der der Authentifizierung dient. Nach der erfolgreicher Authentifizierung und Schlüsselaustausch schaltet er den gesamten Port für beliebigen Verkehr frei.

- **Authentication Server:** Der Authentication Server bekommt vom Supplicant Authentifizierungsanfragen der Supplicants weitergreicht. Er entscheidet ob dem Client Zugriff auf das Netzwerk gestattet werden soll oder nicht.

802.1x ist ein allgemeines Verfahren zur Port basierten Zugangskontrolle. Sein Einsatz ist nicht auf den hier beschriebenen Fall (Wireless LAN's) beschränkt. Es gibt einen Rahmen vor, wie die beteiligten Komponenten den Zugriff auf ein Netzwerk auszuhandeln haben. Verwendet man ein entsprechendes EAP Verfahren (z.B. EAP-TLS) ist ein sicherer Austausch von Schlüsseln zwischen Access Point und Client möglich.

Nach erfolgreicher Authentifizierung eines Clients wird zwischen diesem und dem Authentication Server ein verschlüsselter Tunnel aufgebaut. Der Authentication Server überträgt ein Master Password and den Authenticator. Der kann jetzt die verschlüsselte Sitzung übernehmen, RC4 Schlüssel erzeugen, und diese über den Tunnel and den Client übetragen. Der Tunnel bleibt während der ganzen Session bestehen. Die Komponenten nutzen diesen um während der Session von Zeit zu Zeit neue Schlüssel auszutauschen. Dies erhöht die Sicherheit des Verfahrens, da die Gefahr von so genannten IV Kollisionen (Eine der Hauptschwächen von WEP) verringert wird. Hat bis zu diesem Punkt alles geklappt, öffnet der Authenticator den den Port für den Client

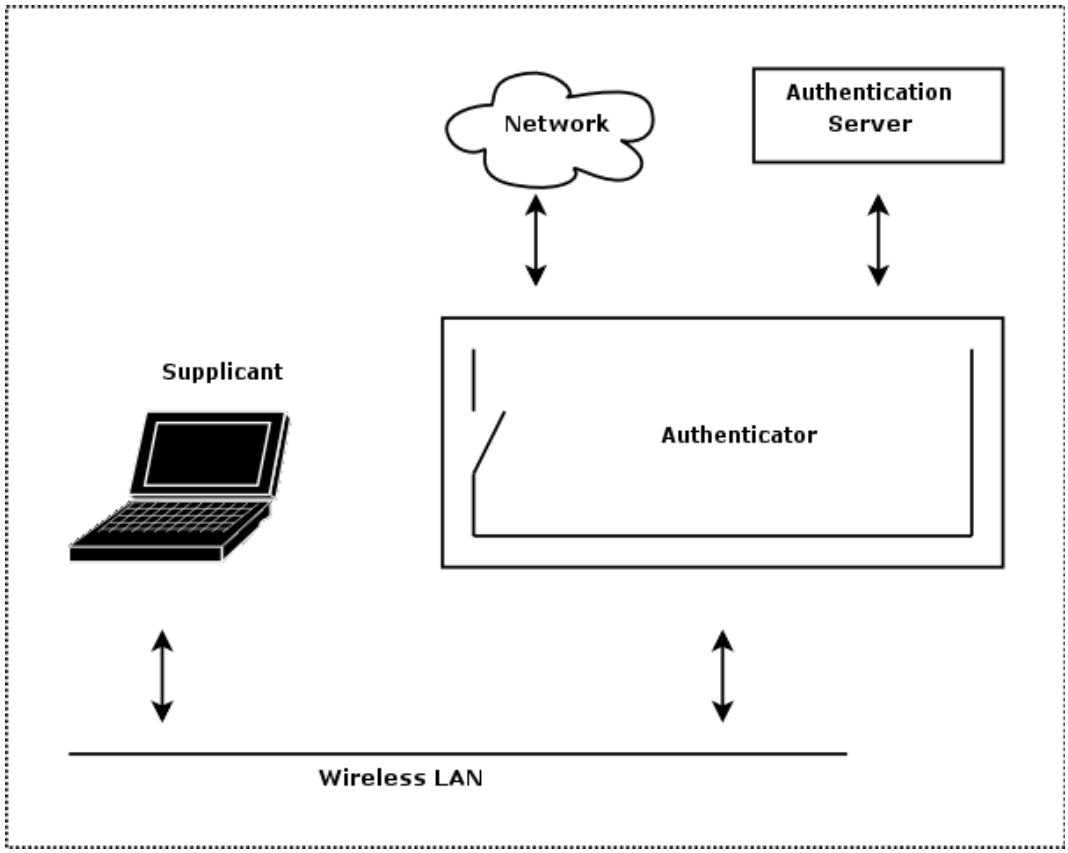


Abbildung 1: 802.1x Übersicht

vollständig. Der Client ist authentifiziert und authorisiert, die Keys für die Verschlüsselung sind ausgetauscht.

Das eben beschriebene Verfahren wird im allgemeinen als WPA-EAP bezeichnet. Verzichtet man auf einen dedizierten Authentication Server, muss das zur Schlüsselgenerierung verwendete Master Password vom Benutzer auf dem Authenticator und dem Supplicant hinterlegt werden. Diese abgerüstete Variante wird als WPA–PSK bezeichnet. Derartige Lösung finden ihre Anwender vor allem in SOHO Bereich. Bei der Wahl des PSK muß man allerdings sehr sorgfältig (besser: zufällig) vorgehen. Schwache Passwörter machen das Verfahren für Wörterbuch Attacken anfällig.

# 2 Implementierung

Ziel dieses Workshops ist es, einen Access Point zu realisieren. der die WPA–Spezifizierung implementiert. Als Platform kommt Linux zum Einsatz, der Authenticator wird durch das HostAP Projekt geliefert, der Authentication Server ist ein freeRADIUS Server. Als EAP Variante soll hier nur EAP-TLS zum Einsatz kommen.

Das HostAP Projekt hat sich (unter anderem) das Ziel gesetzt Accesspoint Funktionalität für Linux basierte Betriebssysteme zu realisieren. Dazu wurden verschiedene Kernel Module für 802.11b Karten mit Prism2/2.5/3 Chipset und User Space Programme entwickelt. Alle Ausführungen beziehen sich auf ein Setup basierend auf einer 802.11b PCI oder Mini-PCI Karte. Solche Karten findet man vereinzelt im Handel oder auf diversen Online Versteigerungsplattformen.

## 2.1 Kernel Konfiguration

Ausgangspunkt für dieses Projekt ist ein Linux Kernel 2.6.7 . Es kann im Prinzip aber jeder aktuelle Kernel aus der 2.6 oder 2.4er Serie verwendet werden. Aus dem HostAP Projekt stammen Kernel Patches, Treibermodule und zusätzliche Programme um Access Point Funktionalität zu realisieren. Hier werden wir die letzte als stabil bezeichnete Version 2.4/2.5 verwenden. Nach Auspacken der Archive an geeigneter Stelle geht es an das Anpassen der Kernel Quellen.

```
bash$>cd linux-2.6.7/
bash$>cat ../hostap-driver-0.2.4/kernel-patches \
/hostap-linux-2.6.2.patch | patch -p1
```

Der HostAP Treiber kommt in 2 seperaten Teilen, als Patch Set für den Kernel und den eigentlichen Treibermoduln. Die Module werden in die Kernel Quellen kopiert.

```
bash$>cd linux-2.6.7/
bash$>cp ../hostap-driver-0.2.4/driver/modules/hostap*.[ch] \
driver/net/wireless/
```

Während der Kernel Konfiguration (`make menuconfig`) sind also die folgenden Eigenschaften zu aktivieren:

```
[*] Wireless LAN drivers (non-hamradio) & Wireless Extensions
[M] Host AP support
[M] Host AP driver for Prism2.5 PCI adaptors
```

Für den Fall das man Besitzer einer Karte mit alter Firmware, ist kann man - wie hier geschehen - noch die Unterstützung für Firmware Downloads aktivieren. Ebenfalls sollte man die benötigten Verschlüsselungsmodule im Kernel aktivieren. HostAP bringt zwar entsprechende Routinen mit; diese werden aber in zukünftigen (fest in den Kernel integrierten) Versionen möglicherweise verschwinden.

```
[M] RC4 cipher algorithm
[M] Michael MIC keyed digest algorithm
```

Voreingestellt ist in der vorliegenden Version die Benutzung der HostAP eigenen Crypto Routinen. Um diese zu ändern begibt man sich nach `drivers/net/wireless` und öffnet die Datei `hostap_config.h` mit einem Editor seiner Wahl. Der Kommentar um die Präprozessoranweisung `#define HOSTAP_USE_CRYPTO_API` muss enfernt werden um den Treiber anzuweisen die Kernel Crypto Module zu verwenden:

```
/* Use Linux crypto API instead of own encryption implementation whenever
 * possible. */
#define HOSTAP_USE_CRYPTO_API
```

Zusätzlich zu diesen HostAP bezogenen Settings aktviert man noch die Option für 802.1d Ethernet Bridiging. Einen Access Point als Bridge zu betreiben ist empfehlenswert, da er so ohne große Änderungen an selbiger in eine bestehende Netztopologie eingegliedert werden kann.

Nach Abschluss dieser Arbeiten kann der Kernel übersetzt und installiert werden. Die Installation folgt den Gegebenheiten von Architektur, Distribution und verwendetem Loader (Grub, lilo , etc.).

## 2.2 Hilfsprogramme übersetzen

Nach hoffentlich erfolgreichem Neustart müssen noch die anderen HostAP Komponenten übersetzt werden.

- `hostap-utils-0.2.4`
- `hostapd-0.2.5`

Dafür sind keine besonderen Vorkehrungen zu treffen. In den Verzeichnissen ist nur `make` aufzurufen; danach sind die Binaries in eine genehme Location (`/usr/local/sbin/`) zu kopieren. Danach ist der richtige Zeitpunkt für eine erste Funktionsprüfung gekommen. Mit `modpobe hostap_pci` lädt man den Kerneltreiber. `iwconfig` sollte zwei neue Netzwerkschnittstellen melden:

```
wifi0     IEEE 802.11b  ESSID:"test"
          Mode:Master  Frequency:2.422 GHz  Access ...
          ...
wlan0     IEEE 802.11b  ESSID:"test"
          Mode:Master  Frequency:2.422 GHz  Access ...
          ...
```

## 2.3 freeRADIUS als Authentication Server

Obwohl der Standard nicht die Verwendung eines bestimmten Authentication Servers vorschreibt wird in aller Regel ein RADIUS Server zum Einsatz kommen. Für die Verwendung von EAP-TLS müssen für den RADIUS Server und für die Supplicants Zertifikate und Schlüssel erzeugt und verwaltet werden. Ebenso ist eine CA notwendig. Man kommt nicht um den Betrieb einer Public Key Infrastruktur herum. Mit `openssl` steht eine freie SSL-Implementierung zur Verfügung, die diese Aufgaben leisten kann. Für die Verwaltung der Server und Client Zertifikate ist der Einsatz einer grafischen Benutzeroberfläche ratsam. TinyCA als Frontend für `openssl` ist dafür gut geeignet. Wer allerdings schon eine PKI im Betrieb hat, sollte diese auch für sein WPA Projekt verwenden.

### 2.3.1 TinyCA, Zertifikate und Keys

Am Anfang einer PKI steht immer die Erstellung eines *Self Signed CA Certificate* Man bedient sich dazu des New CA Buttons aus der TinyCA Toolbar. Das erscheinende Formular füllt man nach bestem Wissen und Gewissen aus. Nach drücken des OK Buttons werden das CA Zertifikat und die zugehörigen Keys generiert. Das Ergebnis sollte ungefähr wie Abbildung 2 aussehen.

Jetzt können die Zertifikat für den Radis Server und den ersten Client generiert werden. Im ersten Schritt lassen wir uns einen Certifikate Signing Request erzeugen. Unter TinyCA muß dazu das Tab mit der Bezeichnung 'Requests' angewählt werden. Danach in der Toolbar 'New' auswählen und das erscheinende Formular ausfüllen. Dabei sollte man beim Zertifikat für den RADIUS Server das Feld *Common Name* mit der eigenen IP Adresse versehen. (Abbildung 3) Beim Client Request trägt man am besten die e-mail Adresse des Zertifikatinhabers ein (Abbildung 4). Ansonsten unterscheiden sich die Requests nicht.

Um die Zertifikate einsetzen zu können, müssen diese noch von der CA unterschreiben werden. Beide Zertifikate sollten nach einem Klick auf das *Requests* Tab im Hauptfenster von TinyCA zu finden sein. (Abbildung 5)

Öffnet man das Kontextmenü für einen Eintrag, sieht man unter anderem 2 Optionen zum signieren des ausgewählten Request. Für den RADIUS Server nutzt man *Sign Request (Server)* und für das Benutzer Zertifikat die mit *Sign Request (Client)* bezeichnete Variante. TinyCA wird nach dem CA Passwort fragen und den ausgewählten Request signieren. Hat TinyCA den Vorgang für beide Requests durchgeführt, sind beide Zertifikate in der Zertifikat Übersicht mit dem Status *valid* aufgegführt. Die Exportfunktion ermöglicht es uns die Zertifikate zu exportieren und sie in den

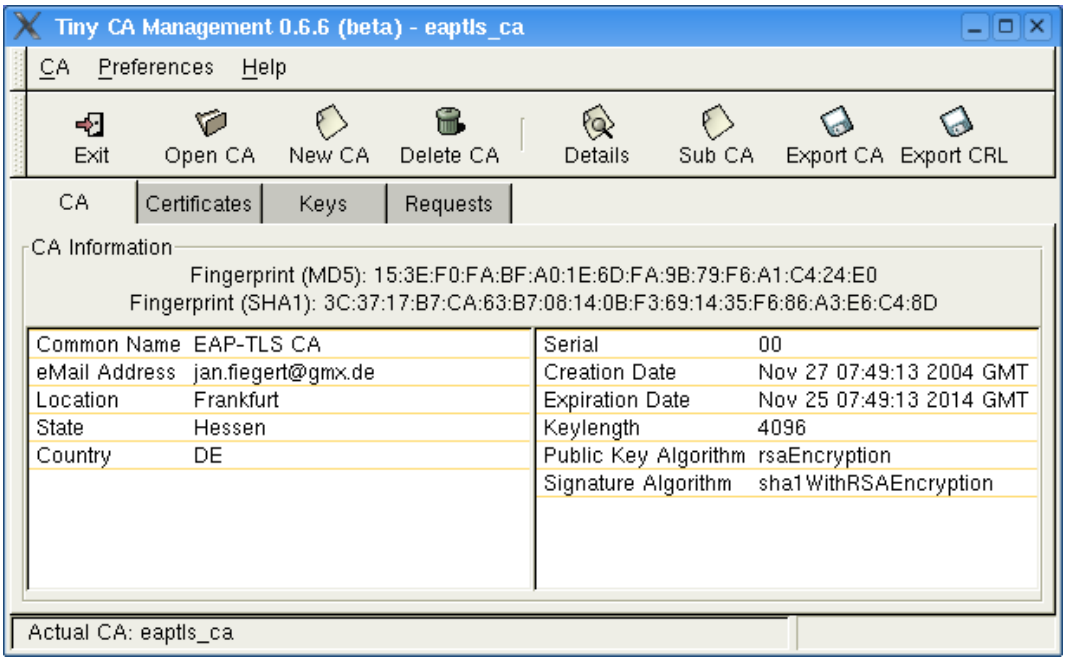


Abbildung 2: TLS Certificate Authority einrichten

Applikationen für die sie gedacht sind zum Einsatz zu bringen. Zum Abschluß lassen wir uns von openssl noch eine Konfiguration für den Diffie-Hellmann Key Exchange Algorithmus generieren:

```
openssl dhparam -check -text -5 512 -out DH
```

### 2.3.2 Konfiguration Radius Server

Die vollständige Konfiugration eines RADIUS Servers hier zu beschreiben würden den Rahmen dieses Workshops sicher sprengen. Ich beschränke mich an dieser Stelle auf die für das EAP-TLS relevanten Settings. Die meisten Distributionen haben schon einige Mühen in die Vereinfachung der Konfiguration gesteckt. So ist die Konfiguration of nach Aufgaben gegliedert in verschiedene Dateien und Vertzeichnisse unterhalb `/etc/raaddb/` aufgeteilt. Um dies schöne Übersichtlichkeit nicht zu zerstören sollte man sich für die für TLS notwendigen zusätzlichen Files ein eigenes Verzeichnis names `certs` anlegen. Hier hinein kopiert man nach erfolgtem Export folgende Files:

- Das Certifikat für den RADIUS Server (`server-cert.pem`)
- Den privaten Schlüssel für den RADIUS Server (`server-key.pem`)
- Das CA Zertifikat (`cacert.pem`)
- Das DH Konfigurations Files (`DH`)

Jetzt kann man sich auf die Suche nach den EAP und TLS bezogenen Settings innerhalb der RADIUS Server Konfiguration machen. Unter dem von mir verwendeten Gentoo-Linux finden sich dies in der Datei `/etc/raddb/eap.conf` Bis auf `random_file = ${raddbdir}/certs/random` sind alle Einträge in der TLS Section bei der Erörterung der *openssl* bereits erläutert wurden. Diese Datei kann durch Eingabe von

```
dd if=/dev/random of=/etc/raddb/random bs=1024 count=3
```

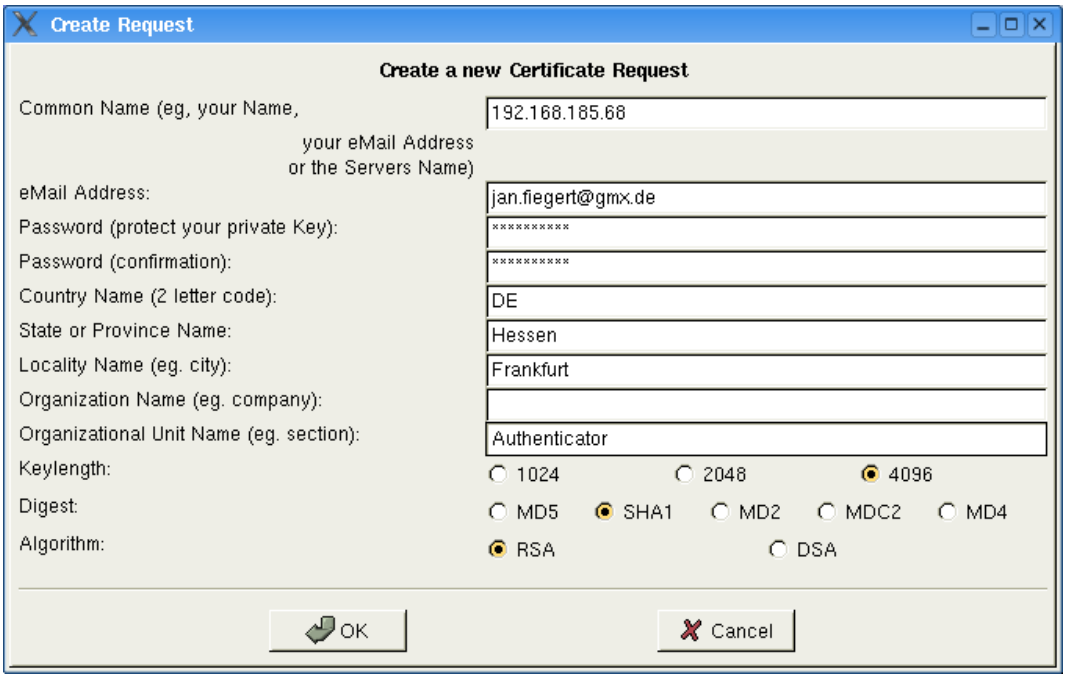


Abbildung 3: Erzeugen eines CSR für den RADIUS Server.

erzeugt werden. Damit der Raidus Server die EAP-TLS Configuration auch verwendet, müssen in der `/etc/raddb/radiusd.conf` eventuell noch Einträge in der `authorize { ... }` und der `authenticate { ... }` Sektionen ergänzt werden. Da der im nächsten Aschnitt beschriebene `hostapd` als Client für den Radius server agieren soll muß er der Radius eigenen Benutzerverwaltung bekannt gemacht werden:

```
client 192.168.185.68 {
        secret          = *****
        shortname       = ap
}
```

Die IP Adresse und das Passwort werden vom Authenticator verwendet um sich beim Radius Server anzumelden. Nach dem Start ist der Radius Server jetzt bereit Anfragen der Supplicants entgegen zu nehmen und dem Authenticator TLS gesicherte Tunnel bereitzustellen.

## 2.4 `hostapd` als Authenticator

Nach diesen vorbereitenden Arbeiten können wir jetzt mit der Konfiguration des `hostapd` beginnen. Da diese Software derzeit noch nicht allzuweit verbreitet, und demzufolge nur wenig Dokumentation im Internet verfügbar ist, wird dies etwas detaillierter als im vorherigen Abschnitt erfolgen.

### 2.4.1 Allgemeine Settings (SSID, Logging, Verschiedenes)

Bis zum Zeitpunkt einer ersten erfolgreichen Verbindungsaufnahme sind ausreichene Debug Ausgaben unbedingt notwendig. Mit den folgenden Anweisungen in der `hostapd.conf` schalten wir die Ausgaben sowohl auf die Console als auch ins Syslog:

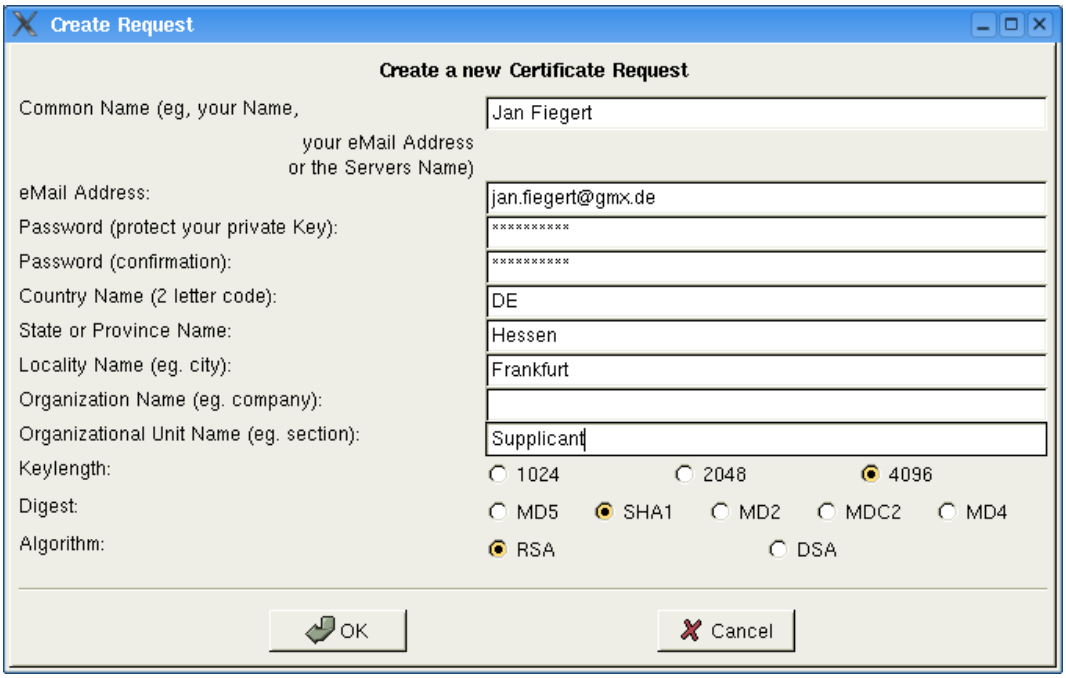


Abbildung 4: Generieren des Request für den Client

```
logger_syslog=-1
logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2
```

Bis alles nach unseren Vorstellungen läuft werden wir den *hostapd* sicher das eine oder andere mal stoppen, die Konfiguration ändern und danch wieder neu starten müssen. Wenn das Programm dabei sich jedesmal in den Hintergrund begibt kann dies auf Dauer lästig werden. Mit der Anweisung

```
daemonize=0
```

unterbindet man dies. Da das Programm jetzt im Vordergrund läuft, kann es bequem mit `CRTL-C` beendet werden. Unter welcher SSID der Accesspoint sich seiner Umwelt announciert legen wir ebenfalls durch einen entsprechenden Eintrag fest.

```
ssid=wpaeap-test
```

### 2.4.2 802.1x und Radius spezifische Settings

Wir folgen dem im Beispiel Konfigurations File gemachten Vorschlag und legen als Authentifizierungs Methode *Open System Authentification* fest:

```
auth_algs=0
```

Die *802.1x* Authentifizierung muß natürlich angeschaltet werden:

```
ieee8021x=1
```

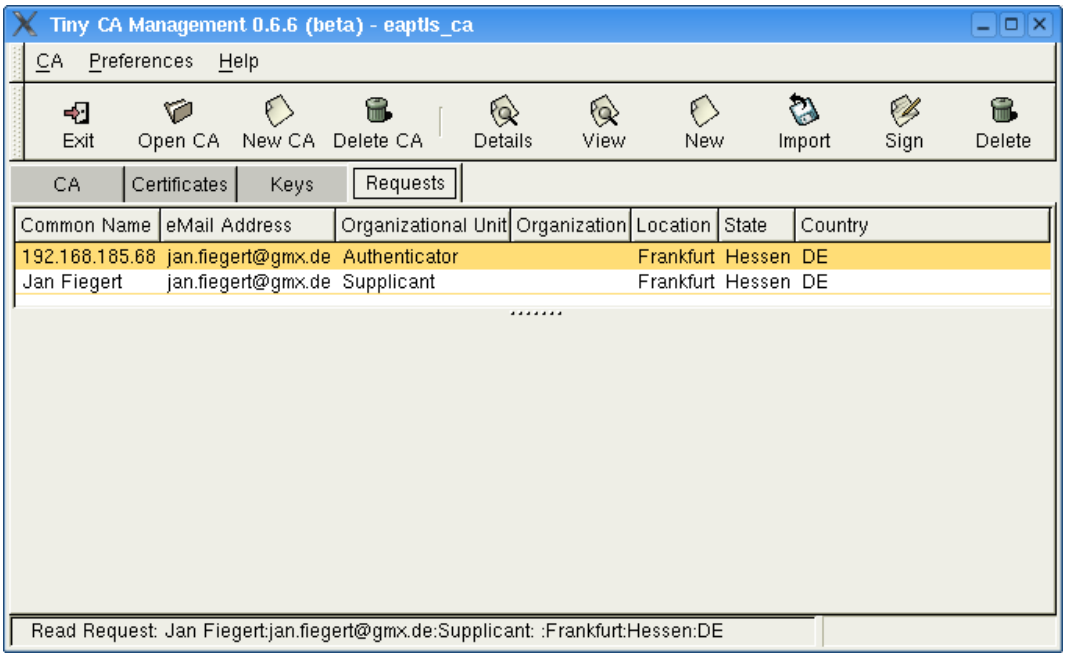


Abbildung 5: Certificate Singing Requests

Der integrierte minimale Authetication Server wird abgeschaltet. Der findet nur bei WPA mit *pre shared keys* Verwendung.

```
minimal_eap=0
```

Mit dieser Adresse identifiziert sich der Access Point beim Radius Server:

```
own_ip_addr=192.168.185.68
```

Die folgenden 3 Einträge beschreiben die Vebindung zum Radius Server.

```
auth_server_addr=192.168.185.68
auth_server_port=1812
auth_server_shared_secret=nunkligh
```

### 2.4.3 WPA Einstellungen

Um die WPA Funktionalität zu aktivieren muss dieser Eintrag auf 1 gesetzt werden:

```
wpa=1
```

`hostapd` unterstützt WPA-PSK und WPA-EAP. Wir aktivieren hier die EAP Funktionalität.

```
wpa_key_mgmt=WPA-EAP
```

Dieser Eintrag gibt die vom Access Point unterstützten Verschlüsselungsalgorithmen vor. Für WPA reicht es aus hier TKIP einzutragen. Dabei handelt es sich um eine ebenfalls auf *RC4* basierende Erweiterung des WEP Verfahrens.

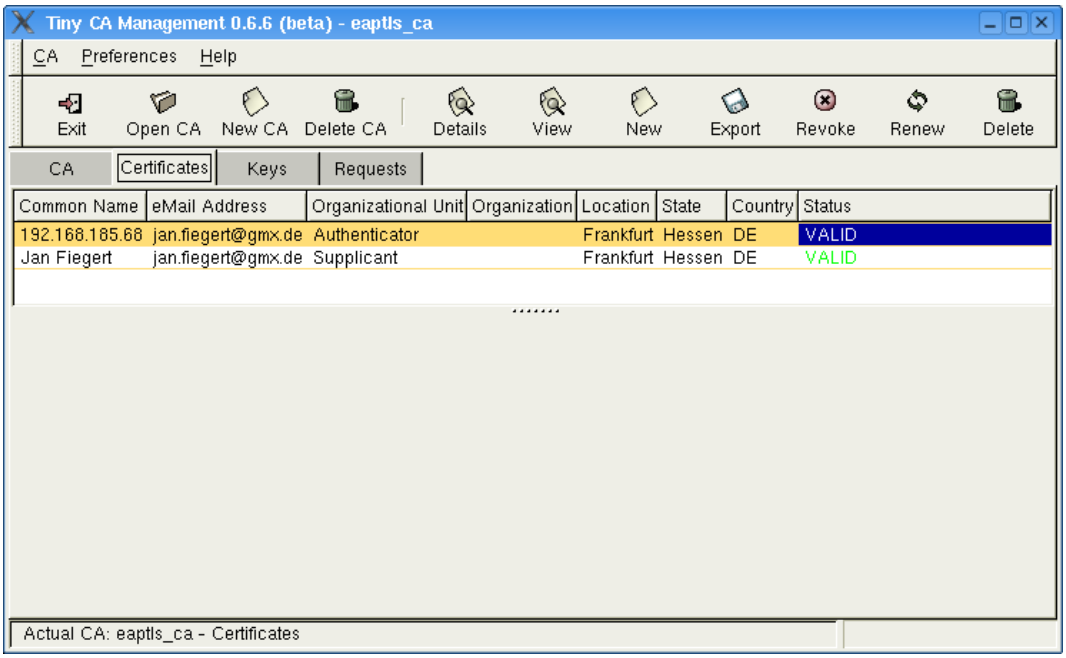


Abbildung 6: Benutzer- und Server Zertifikat

```
wpa_pairwise=TKIP
```

### 2.4.4 Key Management

```
wpa_group_rekey=60
wpa_gmk_rekey=120
```

Diese 2 Timer steuern das gelegentliche Neuaushandeln der für die Kommunikation erforderlichen Keys.

- `wpa_gmk_rekey` steht für die Häufigkeit mir der ein neuer *Master Key* ausgetauscht wird. Der Master Key wird verwendet um das Verschlüsselungsverfahren zu initialisieren.
- `wpa_group_rekey` Nach Ablauf dieses Timers tauschen die Beteiligten einen neuen *Group Key* aus. Der wird verwendet um die Kommunikation an alle Clients (*Broadcast*) zu verschlüsseln.

## 2.5 Inbetriebnahme und Netzwerkkonfiguration

Das Zusammenspiel aller Komponenten jetzt durch den Start des *hostapd* direkt aus der Console oder einem xterm getestet werden:

```
hostapd -dd hostapd.conf
```

Der Radio bezogenen Teil der Konfiguration ist jetzt abgeschlossen. Bleibt noch der IP bezogene Teil Hier stehen prinzipiell 2 Möglichkeiten zur Verfügung.

- **Gateway**
  Das Funkntetz bekommt ein IP Subnetz zugewiesen, die W–LAN Schnittstelle ist das *default*

```
eap {
    default_eap_type = tls
    timer_expire     = 60
    ignore_unknown_eap_types = no
    tls {
          dh_file = ${raddbdir}/certs/DH
          random_file = ${raddbdir}/certs/random
          private_key_password = ******
          private_key_file = ${raddbdir}/certs/server-key.pem
          certificate_file = ${raddbdir}/certs/server-cert.pem
          CA_path = ${raddbdir}/certs/
          CA_file = ${raddbdir}/certs/cacert.pem
     }
}
```

Abbildung 7: freeRADIUS EAP-TLS Konfiguration

```
authorize {
    preprocess
    eap ...
}
authenticate {
    eap ...
}
```

Abbildung 8: `radiusd.conf` (Auszug)

*Gateway* für die Clienten aus diesem Netz. Auf dem Access Point muß man das Routing zwischen dem drahtlosen und dem verdrahteten Netz aktivieren. Diese Lösung ist technisch einfach zu implementieren, aber je nach Umfeld mit einem unter Umständen hohem organisatorischen Aufwand verbunden.

- **Bridge**
  Die Netzwerkschnittstellen werden ohne IP Adressen konfiguriert. Zwischen beiden Netzwerken wird eine Ethernet Bridge etabliert. Die Schnittstelle der Bridge erhält eine Ip Adresse aus dem kabelgebundenen Netz. Funknetz und Ethernet werden zu einem logischen Netz. Der Verwaltungsaufwand für dieses Verfahren ist gering; allerdings ist es technisch etwas aufwändiger zu implementieren. Der Access Point ist aus beiden Netzen unter der Adresse der Bridge zu erreichen. Die Netze befinden sich ebenfalls in der selben Broadcast Domain. Broadcast basierte Dienste wie z.B. Samba werden ohne weiteren Aufwand auch im Funknetz arbeiten.

```
>ifconfig eth0 0.0.0.0 up
>ifconfig wlan0 0.0.0.0 up
>brctl addbr  br0
>brctl addif br0 eth0
>brctl addif br0 wlan0
>ifconfig br0 192.168.185.44 netmask ... up
```

Abbildung 9: Konfiguration Netzwerk Bridge

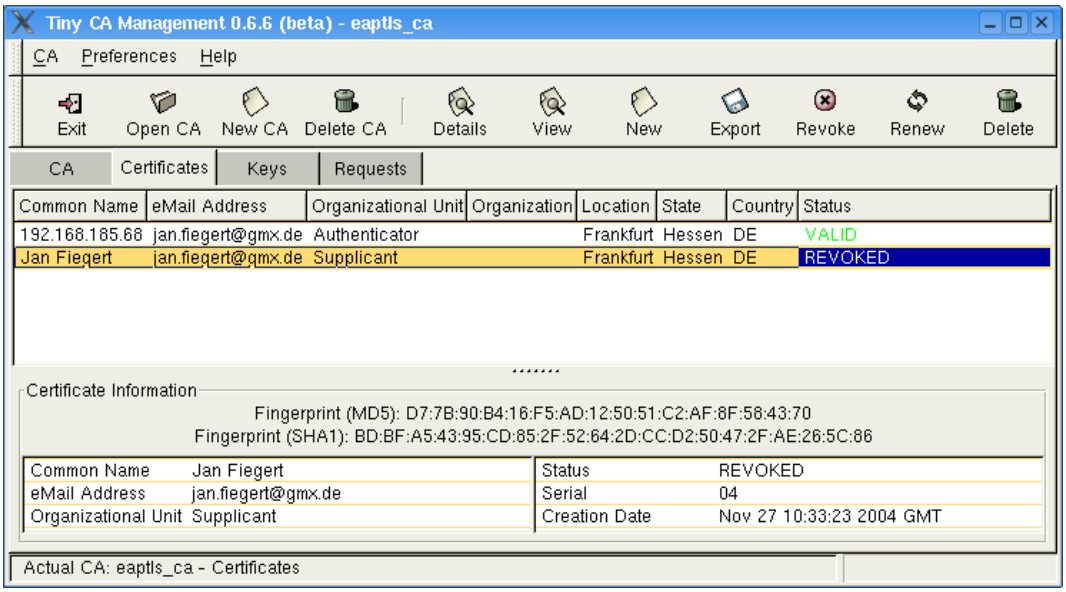


Abbildung 10: Zurückgerufenes Client Zertifikat

Nach diesen abschließenden Arbeiten sollte man sich mit WPA fähigen Endgeräten am Eigenbau Access Point anmelden.

## 2.6 CRL basiertes Benutzermanagement

Benutzerverwaltung ist natürlich keine Einbahnstraße. Von Zeit zu Zeit sollen Bentuzer auch wieder vom Access Point ausgeschlossen werden. Dafür muß zunächst sein Zertifikat wiederrufen, und eine Certificate Revokation List erstellt werden. Die Funktion zum exportieren der CRL erreicht man über die Toolbar nach klicken auf das CA Tab. ( Abbildung: 10) Die Konfiguration des Radiusservers ist in der EAP-TLS Sektion um die folgenden beiden Einträge zu erweitern

```
CA_path = ${raddbdir}/certs/
check_crl = yes
```

Die CRL kopiert man in das durch `CA_path` bezeichnete Verzeichnis. Nach Aussführen von

```
c_rehash your CA_path
```

und anschliesendem Neustart des Radiusservers prüft dieser die CRL und weisst gegebenenfalls ungültige Zertifikate zurück.

# 3 Links

http://hostap.epitest.fi – HostAP Projekt Seite
http://www.kernel.org – Linux Kernel
http://www.freeradius.org – freeRADIUS Server Projekt