Recommended Security System for wireless networks

Implementation of IEEE 802.1X

Best Practice Document

Produced by UNINETT led working group on mobility
(No UFS112)

Authors: Jardar Leira
May 2010

Document No: GN3-NA3-T4-UFS112
Version / date: May 2010
Original language: Norwegian
Original title: “UFS 112: Anbefalt sikkerhetsløsning for trådløse nettverk. Implementasjon av IEEE802.1X”
Original version / date: Revision 1 of 20 December 2007
Contact: campus@uninett.no

UNINETT bears responsibility for the content of this document. The work has been carried out by a UNINETT led working group on physical infrastructure as part of a joint-venture project within the HE sector in Norway.

This translated version is based on the Norwegian counterpart approved by the Norwegian HE sector on 20 December 2007 after an open consultation period of four weeks.

The translation of this report has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°238875, rel ating to the project 'Multi-Gigabit European Research and Education Network and Associated Services (GN3)'.

Table of Contents

Executive Summary

This document provides information about the different security mechanisms available for wireless networks. It describes the shortcomings of using MAC address filters, WEP, Web portals and VPN and recommends mutual authentication based on 802.1X as the best alternative. EAP using TLS, PEAP and TTLS are recommended alternatives which can also be supported simultaneously by the system. The IEEE 802.11i standard is better known as WPA and WPA2 with TKIP and AES encryption, respectively. AES encryption is preferred but TKIP is a compromise in the event of clients lacking support for AES. Familiarising oneself with certificate generation can be complicated but this document contains scripts which can simplify the process. Eduroam is an international RADIUS hierarchy for academic institutions which makes it possible for users to log in to the wireless networks of other member institutions using their own user names and passwords.

Introduction

This document provides specification of the Norwegian HE sector's recommended solution security system for wireless networks. The solution uses IEEE 802.1X. This translated document is on the document approved by the Norwegian HE sector on 20 December 2007 after an open consultation period of four weeks.

The target group comprises IT managers and IT operations personnel in the HE sector.

1 Background

A wireless network is fundamentally far less secure than a cable network. The reason for this is that wireless networks are based on the use of radio waves and by their very nature broadcast any data being transmitted between clients and access points. As a result, anybody within range can listen to any communications. The use of a sensitive antenna will considerably extend the range of such eavesdropping. The absence of a cable between the client and the access point also presents a security challenge since an access point's identity (SSID and BSSID) can easily be forged, as they are not intended to be used as security measures. It is therefore not possible for the client to be sure that the access point it is connected to can be trusted, which may have consequences for the further transmission of data. On the other hand, the access point cannot be sure that the customer is a valid user either, since all that is required from the client's side is a correct SSID.

To summarise, we can say that wireless networks present two security challenges:

1. Secure, mutual authentication of access point and client
2. Security of data communication against eavesdropping and forgery.

Based on this background we have evaluated several different methods for making wireless networks more secure.

- **MAC address lists** – Only clients with approved, registered MAC addresses are allowed to associate with access points.
  This is very labour-demanding in its management and provides no security advantages since MAC addresses are easy to discover and forge. Neither does it provide possibilities for encryption of data communications.
- **WEP encryption** – Available with different levels of encryption (RC4). A common, shared secret must be known in order to associate with an access point and also enables encryption of the data communication.
  WEP has several weaknesses and can today be cracked in less than a minute using software easily obtainable on the Internet. A shared secret does not scale well and the data are encrypted in such a way that anybody with knowledge of the key can listen to all data communication.
- **Web portals** – Clients connect to an unencrypted, open wireless network. When a user tries to use a web browser to access an external web page, the request is redirected to a web-based login page for the network. The page is usually protected using communication via SSL.
  This is a user-friendly solution which is often used in "hot spots" and publicly accessible wireless networks. Mutual authentication of the system and client takes place via SSL, but in practice it is too easy for a user to ignore warnings of errors in certificates or if SSL is being used at all. Because there is no verification of the access point (the access point), this method is very vulnerable to man-in-the-middle attacks and phishing. Neither is the data communication encrypted. One should generally be very cautious when using such systems.
- **VPN** – An otherwise completely open wireless network is defined as an "insecure external network" and traffic is then only permitted between a concentrator and wireless clients. Clients must authenticate themselves on the VPN concentrator in order to create an encrypted tunnel within the wireless network. This method provides high security with mutual authentication and individual encryption of data communication. However it makes heavy demands on the processing capacity of the concentrator, since all wireless users will have to use it. Client software leads to further complications, resulting in considerable demand for user support, and some operating systems may experience software limitations.

Shortly after it became known to the world's wireless users that WEP had been cracked, there was considerable demand for the introduction of a simple and effective system for securing wireless networks. By this time the IEEE had already made good progress towards developing the 802.11i standard and this was to provide a system which dealt with both security-related challenges.

2 IEEE 802.11i with IEEE 802.1X, TKIP and AES

IEEE 802.11i is our recommended security solution for wireless networks. We provide a description of this standard below.

2.1 802.11i and WPA/WPA2

With the IEEE 802.11i standard we now have the ability to use IEEE 802.1X for authentication, also in wireless networks. While the 802.11i standard was being developed, the Wi-Fi Alliance launched a system they called WPA. This uses 802.1X for authentication and TKIP for encryption. When 802.11i was complete, WPA2 was introduced, which is 802.1X with AES encryption. 802.11i is not as familiar to users as WPA and WPA2, but WPA and WPA2 can be said to represent the Wi-Fi Alliance's certification of products which have implemented 802.11i in a way which makes them compatible with other WPA-certified products. In other words, WPA/WPA2 certification of a product shall ensure that the user has equipment which will function in combination with other manufacturers' products.

There are two versions of both WPA and WPA2: "Enterprise" and "PSK" (Pre-Shared Key). Enterprise uses 802.1X and is suitable for corporate networks while PSK uses a shared secret of 64 hexadecimal characters and is intended for use by small businesses or at home (SoHo).

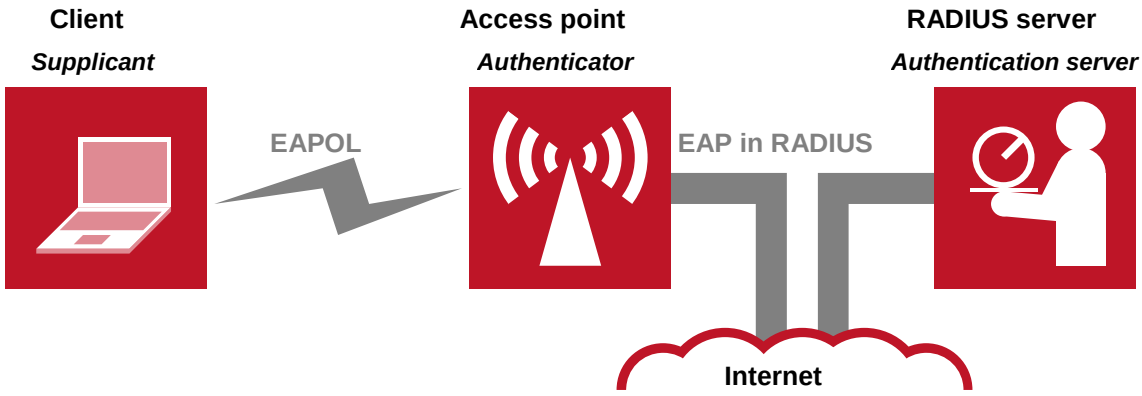
PSK is basically not suitable for use in academic institutions, but since very many users have a wireless access point at home, some information about PSK is provided in this document for security reasons. Although WPA with PSK is a significant improvement over WEP, it is important to select a password or pass phrase with care so that it is not too short or easy to guess. Allowing users to fill in all 64 hexadecimal characters themselves is not a very user-friendly arrangement. Hence the manufacturers have implemented an algorithm which expands a password or pass phrase to 64 hexadecimal characters. If this password or pass phrase is less than 20 characters, one has in reality not achieved much higher security than if one used WEP. Software is available on the Internet which can crack such short WPA-PSK passwords and pass phrases.

Note that the descriptions below focus on recommended or commonly-used systems to achieve optimal security. Unfortunately, as is so often the case, it is possible to configure even WPA/WPA2 in ways which make them far less secure.

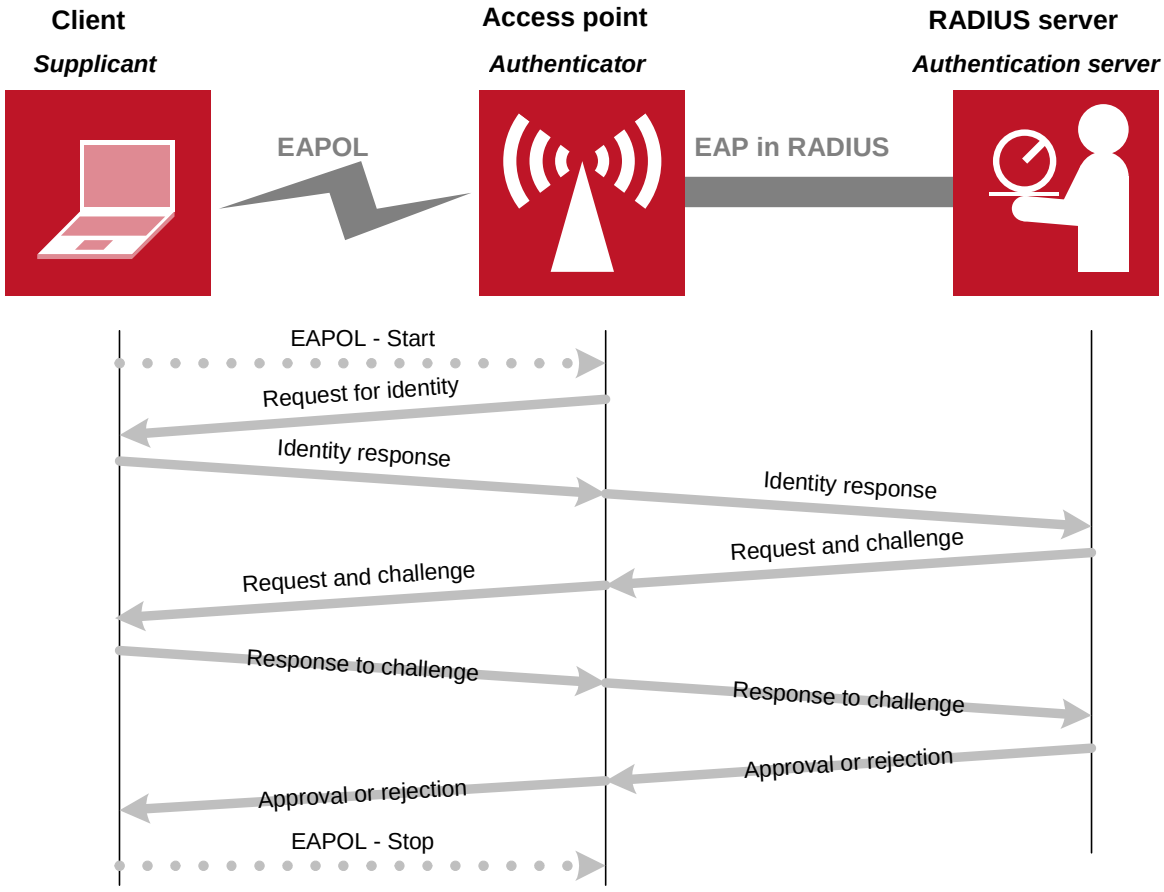
2.2 The elements of 802.1X

The core of IEEE 802.11i is IEEE 802.1X. IEEE 802.1X makes use of three elements in an authentication process: Supplicant (the client), Authenticator (AP) and Authentication Server (RADIUS). The communication between these takes place via a protocol called EAP (Extensible Authentication Protocol). EAP between the

Supplicant and Authenticator works directly on Layer 2 (EAPOL), while between the Authenticator and the Authentication Server it functions via TCP/IP as part of the RADIUS protocol.

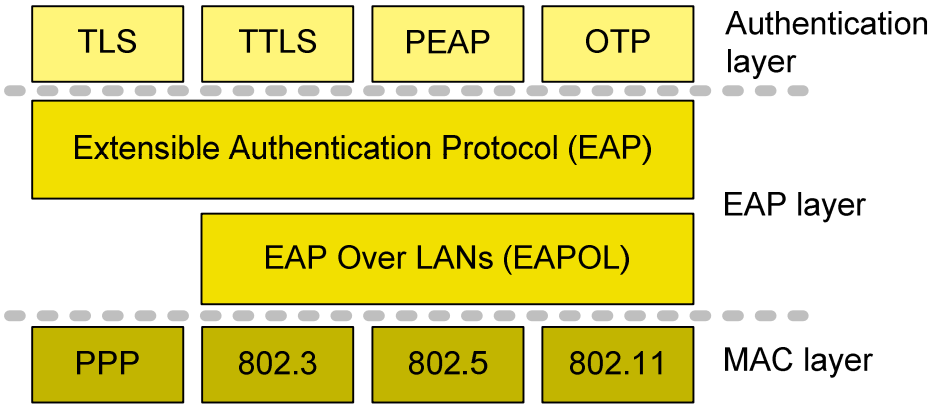


The authentication process is initiated by either the Supplicant or the Authenticator when the client tries to associate with the access point. There are several ways of doing this, but in the case of mutual authentication the Authentication Server will first send its certificate to the Supplicant via the Authenticator. If the Supplicant accepts this, it will respond by providing its user name and password or certificate, depending on which method is in use. If the Authentication Server accepts this, it will send acknowledgement to the Authenticator which will then open a wireless “port” for the Supplicant. This virtual port is a secure wireless connection between the Supplicant and the Authenticator enabling the client to obtain open access to the network behind the access point.



The figure above illustrates the EAP process. The initial process in which the RADIUS server first sends its public key is not included in this figure, but will take place before the client is asked to identify itself.

EAP is constructed to enable the use of an optional authentication mechanism.



Many alternatives are available but few of these provide the mutual authentication we need in wireless networks. These are TLS (Transport Layered Security), TTLS (Tunneled TLS) and PEAP (Protected EAP). What all of these have in common is that the Authentication Server must have a server certificate. TLS also uses a client certificate for each user while TTLS and PEAP, which are very similar, make use of a user name and password. TTLS and PEAP should hence use MS-CHAPv2 to package the credentials before sending them to the Authentication Server. There is no technical reason why several authentication methods should not be offered simultaneously. Many operators running 802.1X for wireless networks give the client a choice between TLS, PEAP and TTLS.

2.3 Certificates and user authentication

Since these methods involve the use of certificates, a PKI system must be set up as a minimum requirement. This means that one must either use an existing Certificate Authority or one must set up one's own. On a SAMSON 3 (or other Linux servers), this is fairly trivial (see separate section), as is also the case when using Microsoft IAS or NPS. Setting up one's own CA is free of charge and one can generate as many server and user certificates as one wishes. The drawback is that one's own CA will initially be unknown to all your users, so that they will be unable to verify the authenticity of the certificate received from the Authentication Server. In order to do this, all clients must install the public certificate of the local CA. This can safely be distributed by way of e-mail or websites, but this is an additional operation in the configuration which the user must perform correctly.

Alternatively, a server certificate may be purchased from an organisation which has pre-installed the public certificate of its CA in the client's operating system. There are many alternatives to choose from and many different pricing models. Whether this will be financially worthwhile will depend on weighing the cost of user support against the cost of the user certificate.

It should also be mentioned that using certificates from a large CA **may** involve a security risk. In practice, very few clients will check whether certificates belong to another server with the right name, but will be content with checking that the certificate is valid. Hence one may theoretically be presented with a valid certificate by a server which has been configured for phishing. At present the risk of this happening is considered rather small.

When it comes to the identification of a user or client, certificates may also be used. Each user receives a personal certificate and a computer can be given host specific certificates. In this scenario, users will not need to have a user name and password. On the other hand, the organisation must have a good PKI system, with certificate handling, which can be expensive in terms of both cost and management resources.

The alternative is to use user names and passwords. This method is considered as secure as using certificates and has the added advantage of enabling the linkage of authorisation to an existing user database. However, it is a precondition that the user database can check MD4-hash passwords, as this is the format of passwords in MS-CHAPv2.

2.4 Encryption

It should be noted that when 802.1X authentication is used, only the authentication process is encrypted. Following the completion of authentication, data communication can take place without encryption.

At the same time, using 802.1X makes it possible for access points to distribute, in a secure manner, a unique per-user key which can be used for encryption. This is used in WPA by employing TKIP encryption and in WPA2 with AES encryption. This provides an effective and unique per-packet encryption of all data traffic. It is possible to configure a wireless network with 802.1X security so that it does not make use of encryption or use rotating WEP keys. Although this is far more secure than ordinary WEP if frequent rotation is used, it is not recommended.

TKIP is an RC4 encryption like WEP but thanks to some powerful improvements it does not have the same shortcomings. Older wireless cards which supported 128 bit WEP may be firmware upgraded to run TKIP but unfortunately several manufacturers have not prioritised this in older models. TKIP is not completely secure and has some shortcomings but is nevertheless generally considered very secure, so far.

AES is very similar to Rijndael, which is the “original”. AES has some limitations in, for example, its key size. It is used by governmental organisations in the USA, being used for the encryption of classified material up to the level of TOP SECRET. In other words it is considered to be very secure.

3 Certificate generation on a SAMSON 3 server (Linux)

The generation and use of certificates can, as is the case with so many other things, seem confusing, complicated and laborious before one has familiarised oneself with it. The following description attempts to explain the process in a simple manner. Finally, three scripts are presented which automate much of these manual operations.

3.1 Creating a root CA certificate and setting up certificate hierarchy

When setting up one's own certificate hierarchy, one must first establish a root certificate which will later be used to generate and control all other certificates.

```
openssl req -new -x509 \
            -keyout root-req.pem \
            -out root-req.pem \
            -days 1825 \
             -passout pass:capassword
```

This generates a new X509 certificate and a private key for the same file in PEM (Privacy Enhanced Mail) format. This example results in a CA with a duration of 1825 days, or five years. There is nothing to prevent one from creating a certificate with longer or shorter duration, but since all other certificates are controlled by this one, the chosen duration should be so long that they do not have to be changed inconveniently frequently. The duration is often set to as much as ten years.

You will be asked to enter some information:

```
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:
Email Address []:
```

Enter the information in a way that seems reasonable to you. Take care to make the "Common Name" (CN) a simple name such as "UNINETT WLAN CA", since this is the name that all users will see in their list when they use the certificate.

Instead of entering all this information manually when generating a certificate, one may edit the configuration file for OpenSSL `/etc/ssl/openssl.cnf`. Some way down the file text you will find entries such as *`countryName_default, organizationalUnitName_default`*, and so on. Enter the standard values that you wish to be displayed. You may also enter *`_default`* for entries that are missing. The result of this operation will be a file called "`root-req.pem`".

You must now set up a certificate hierarchy, using the *`CA.pl`* script which is included in OpenSSL, and the `-newca` parameter.

```
/usr/lib/ssl/misc/CA.pl -newca
```

At the prompt: `CA certificate filename (or enter to create)`
Enter the name of the certificate we have just generated: root-req.pem

Depending on what has been configured in `openssl.cnf`, a catalogue will be created (demoCA) with an underlying catalogue structure and files which will contain the certificate hierarchy. The public certificate for the CA is also located here.

In addition we need a file which keeps track of the serial number of the next certificate which will be created. The simplest way to do this is:

```
echo "01" > demoCA/serial
```

Verify that `demoCA/cacert.pem` contains the same certificate as that located in *`root-req.pem`* and that *`demoCA/private/cakey.pem`* contains the same key as `root-req.pem`. Then you can delete `root-req.pem`.

```
rm root-req.pem
```

Finally, you must create a public certificate in DER (Distinguished Encoding Rules) format to enable users to import it to their clients as the root CA.

```
openssl x509 -inform PEM \
             -outform DER \
             -in demoCA/cacert.pem \
             -out CAroot.der
```

The `CAroot.der` file can be given a suitable name and distributed freely to any of one's own users who are likely to use the wireless network. It may be sent, for example, by e-mail, inserted on a website or be available at a reception desk on a memory stick. In Windows it is imported simply by double-clicking on it.

3.2 Creating a server certificate

The server certificate is necessary irrespective of whether one is to use TLS, PEAP or TTLS authentication because it is the only way in which the RADIUS server can identify itself.

The first thing we must do is to request a file in PEM format.

```
openssl req -new \
            -keyout req-server.pem \
            -out req-server.pem \
            -days 1825 \
            -passout pass:serverpassword
```

As when we created the CA certificate, we are asked to enter values for the organisation, location and name. Enter what seems appropriate. It is sensible to let the duration of the server certificate be a few years, but there is no problem in replacing the certificate when necessary and this has little or no effect on the clients' configuration.

IMPORTANT! The Common Name (CN) must be the full DNS name (host name plus domain name) of the RADIUS server, for example radius.uninett.no. This is because some clients will check whether the CN in the RADIUS server agrees with what they expect it to be. Since the client naturally has no possibility of checking directly in the DNS during the authentication process, the user must enter this himself. For many users it will be natural to use the name assigned by the DNS.

The following questions are also asked:

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

These fields should simply be left empty.

The command results in the creation of the file *req-server.pem*.

This must be signed by the CA and have the necessary policy added so that it can be used for our intended purposes. The different purposes for which a certificate may be used are specified with the help of an OID (Object ID). We must specify that this certificate will be used by a server for identification. For this the OID is 1.3.6.1.5.5.7.3.1, which we for the sake of simplicity enter into a file called *EXTENSIONS*.

```
[ xpserver_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
```

The certificate is signed as follows:

```
openssl ca  -policy policy_anything \
             -out server-cert.pem \
             -extensions xpserver_ext \
             -extfile EXTENSIONS \
             -passin pass:serverpassword \
             -key capassword \
             -infiles req-server.pem
```

The result is the file *server-cert.pem* which we will use first to generate the certificate in P12 format:

```
openssl pkcs12 -export \
                -in server-cert.pem \
                -inkey req-server.pem \
                -out server.p12 \
                -clcerts \
                -passin pass:serverpassword \
                -passout pass:serverpassword
```

The file `server.p12` which is created is then used to create a final certificate in PEM format:

```
openssl pkcs12 -in server.p12 \
                -out server.pem \
                -passin pass:serverpassword \
                -passout pass:serverpassword
```

Strictly speaking, the only file you need for the RADIUS server is *server.pem*, which means that you may now delete `req-server.pem`, `server-cert.pem` and `server.p12`, unless you have a RADIUS server which requires a certificate in P12 format.

```
rm req-server.pem
rm server-cert.pem
rm server.p12
```

3.3 Creating a client certificate

If you are to run a TLS system, you must also issue client certificates. Creating a client certificate is only marginally different from creating a server certificate.

The first thing we must do is to request a file in PEM format.

```
openssl req -new \
             -keyout req-client.pem \
             -out req-client.pem \
             -days 1095 \
             -passout pass:clientpassword
```

We are asked to enter values for the organisation, location and name. Enter what seems appropriate. It may be practical to set the validity of the certificate to the duration of the course or employment contract, for example 3 years.

IMPORTANT! It is recommended that the Common Name (CN) be the user's user name plus home organisation or domain, e.g. user@uninett.no, to enable correct routing of the query through a RADIUS hierarchy (see the section about eduroam).

The following questions are also asked:

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

These fields should simply be left empty.

The command results in the creation of the file *req-client.pem*.

This must be signed by the CA and have the necessary policy added so that it can be used for our intended purposes. The different purposes for which a certificate may be used are specified with the help of an OID (Object ID). We must specify that this certificate will be used by a client for identification. For this the OID is 1.3.6.1.5.5.7.3.2, which we for the sake of simplicity enter into a file called *EXTENSIONS*.

```
[ xpclient_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2
```

The certificate is signed as follows:

```
openssl ca  -policy policy_anything \
             -out client-cert.pem \
             -extensions xpserver_ext \
             -extfile EXTENSIONS \
             -passin pass:clientpassword \
             -key capassword \
             -infiles req-client.pem
```

The result is the file *client-cert.pem* which we will use first to generate the certificate in P12 format:

```
openssl pkcs12 -export \
                -in client-cert.pem \
                -inkey req-client.pem \
                -out client.p12 \
                -clcerts \
                -passin pass:clientpassword \
                -passout pass:clientpassword
```

The `client.p12` file is the format the majority of clients can use. For a Windows client it is only necessary to double-click the file to import it to the correct location. The “clientpassword” entered here is the user’s password for the private key and must be used to import the file to the user’s own computer.

You may now delete the files `req-client.pem` and `client-cert.pem`.

```
rm req-client.pem
rm client-cert.pem
```

3.4 Scripts for generating certificates

ca-cert.sh

```
#!/bin/sh
PATH=/etc/ssl/misc/:${PATH}
CA_PL=/usr/lib/ssl/misc/CA.pl

if [ $# != 1 ]
then
        echo "Format: $0 capassword"
        exit 1
fi

# If the CA hierarchy already exists, the earlier structure must
# be deleted. Otherwise the new CA certificate will not be registered.
rm root*
rm -rf demoCA

# Create a new CA root certificate
openssl req -new -x509 -keyout root-req.pem -out root-req.pem -days 1825 \
-passout pass:$1

# Create CA hierarchy
echo "root-req.pem" | $CA_PL -newca >/dev/null

# Create serial number file
echo "01" >> demoCA/serial

# Create CA certificate in DER-format
openssl x509 -inform PEM -outform DER -in demoCA/cacert.pem -out CAroot.der

# Tidy up
rm root-req.pem
```

EXTENSIONS

```
[ xpclient_ext]
extendedKeyUsage = 1.3.6.1.5.5.7.3.2

[ xpserver_ext ]
extendedKeyUsage = 1.3.6.1.5.5.7.3.1
```

server-cert.sh

```
#!/bin/bash

if [ $# != 3 ]
then
        echo "Format: $0 file name CA-password certificate-password"
        exit 1
fi

# Request a new certificate
openssl req -new -keyout req-$1.pem -out req-$1.pem -days 1825 \
-passout pass:$3

# Sign certificate and enter the necessary OID
openssl ca  -policy policy_anything -out $1-cert.pem \
-extensions xpserver_ext -extfile EXTENSIONS \
-passin pass:$3 -key $2 -infiles req-$1.pem

# Export in P12 format
openssl pkcs12 -export -in $1-cert.pem -inkey req-$1.pem -out $1.p12 \
-clcerts -passin pass:$3 -passout pass:$3

# Convert to PEM format
openssl pkcs12 -in $1.p12 -out $1.pem -passin pass:$3 -passout pass:$3

# Tidy up
rm req-$1.pem
rm $1-cert.pem
```

client-cert.sh

```
#!/bin/bash

if [ $# != 3 ]
then
        echo "Format: $0 file name CA-password certificate-password"
        exit 1
fi

# Request a new certificate
openssl req -new -keyout req-$1.pem -out req-$1.pem -days 1095 \
-passout pass:$3

# Sign certificate and enter the necessary OID
openssl ca  -policy policy_anything -out $1-cert.pem \
-extensions xpclient_ext -extfile EXTENSIONS \
-passin pass:$3 -key $2 -infiles req-$1.pem

# Export in P12 format
openssl pkcs12 -export -in $1-cert.pem -inkey req-$1.pem -out $1.p12 \
-clcerts -passin pass:$3 -passout pass:$3

# Tidy up
rm req-$1.pem
rm $1-cert.pem
```

4 Purchasing a certificate from UNINETT SCS

The alternative to creating certificates is to purchase them from an established certificate issuer. The advantage of this is that the CA is already installed in the client computers. Users wishing to purchase certificates can do so from any supplier they wish to.

UNINETT provides a Server Certificate Service (SCS) as a result of its membership of TERENA's SCS project. The certificates are issued by GlobalSign, a CA located in Belgium which belongs to the Cybertrust group.

The necessary information and procedure for ordering will be found at http://forskningsnett.uninett.no/scs/.

Briefly, the procedure is as follows:

1. Your organisation must have arranged Pre-approval
2. Generate key pairs and a Certificate Signing Request (CSR)
3. Submit the CSR via the SCS enrolment pages
4. Confirm the order
5. Receive the certificate
6. Install the certificate

Experience has shown that generating a CSR can be difficult from certain platforms or applications. We therefore recommend OpenSSL, which is available for both Windows and Linux.

5 Configuration of FreeRADIUS on a SAMSON 3 server (Linux)

FreeRADIUS is based on open source code and is in widespread use. Initially, the configuration files, and particularly the debug messages, of the program may seem incomprehensible, but with a little experience it is fairly simple to use. It can be associated with a number of different types of user databases, as with PAM, UNIX, SQL and LDAP. The most commonly used user database is probably Active Directory (AD).

For users wishing to install this themselves on their own Linux-based servers, either in the form of binary distribution or from locally compiled source code, it will be time-saving to know that the Debian distribution is not compiled with the necessary EAP support. The standard *configure* in the source code does not include the necessary parameters for this, either. The installation on SAMSON 3 servers has however corrected this.

In order to compile with the necessary EAP support, one must when using *configure* at least run:

```
./configure --with-rlm_eap_tls --with-rlm_eap_peap --with-rlm_eap_ttls
```

This will provide the necessary support for TLS, PEAP and TTLS.

All the configuration files for FreeRADIUS will normally be found under /etc/freeradius. There are rather a lot of them, but the ones which are important for those of us with 802.1X are:

- radiusd.conf – the main configuration file
- eap.conf – all the parameters for authentication via EAP.
- clients.conf – a list of access points or controllers which are permitted to use RADIUS. There is also a list of other RADIUS servers which are permitted to make contact.
- proxy.conf – a list of where the different realms shall be handled, i.e. which RADIUS servers shall handle a query.
- users – user names and passwords in text format and/or special treatment of authentication if a user satisfies certain conditions. The *users* file is not recommended for use as a user database.

All the configuration files are accompanied by a standard set-up. In this document we only deal with the elements which are necessary to get 802.1X operating with TLS, PEAP and TTLS. Local adaptations for special authentication with reference to a given database will come in addition.

radiusd.conf

```
#Receive RADIUS queries via other RADIUS servers
proxy_requests  = yes

#Include configuration file for RADIUS proxy
$INCLUDE  ${confdir}/proxy.conf

#Include configuration file for RADIUS clients
$INCLUDE  ${confdir}/clients.conf

#Section with definition of modules
modules {
       #Include configuration file for EAP
       $INCLUDE  ${confdir}/eap.conf

       #Enable use of MS-CHAP (and MS-CHAPv2)
       mschap {
       authtype = MS-CHAP
       }

       #Define suffix and delimiter @
       realm suffix {
              format = suffix
              delimiter = "@"
              ignore_default = no
              ignore_null = no
       }
       #Use “users” file
       files {
              usersfile = ${confdir}/users
       }
}

# Which modules shall be used for authorisation.
authorize {
        suffix
        eap
        files
}
# Which modules shall handle the different authentication methods.
authenticate {
       #MS-CHAP for MS-CHAPv2 in PEAP and TTLS
       Auth-Type MS-CHAP {
              mschap
       }
       #EAP messages
       eap
}
```

eap.conf

```
eap {
       # Select expected standard EAP type (tls, peap or ttls)
       default_eap_type = peap
       # Do not allow too long time between response and query.
       timer_expire = 60
       # Handle unknown/unconfigured EAP types? no/yes
       ignore_unknown_eap_types = yes
       # There is an error in Cisco AP1230B firmware 12.2(13)JA1
       cisco_accounting_username_bug = no

       # Handling of EAP-TLS. Also server certificate for PEAP and TTLS.
       tls {
       # Password for RADIUS server's private key
              private_key_password = hemmelig
              # Private key and certificate can be located in same file
              # or in two separate files.
              private_key_file = /etc/freeradius/certs/server.pem
              certificate_file = /etc/freeradius/certs/server.pem
              # Public certificate for CA
              CA_file = /etc/freeradius/certs/demoCA/cacert.pem
              # DH file
              dh_file = /etc/freeradius/certs/dh
              # File with random content (random seed)
              random_file = /etc/freeradius/certs/random
              # Most APs have maximum packet size of 1500-1600 bytes
              # In which case this value must be set at 1024 or less.
              fragment_size = 1024
       }

       # Handling of EAP-TTLS
       ttls {
       # Recommended standard EAP type in TTLS
              default_eap_type = mschapv2
       }

       # Handling of EAP-PEAP
       peap {
       # Recommended standard EAP type in PEAP
              default_eap_type = mschapv2
       }

       # Handling of EAP-MS-CHAPv2
       mschapv2 {
       }
}
```

clients.conf

```
# IP address of Authenticator, i.e. access point/controller which shall
# use the RADIUS server for client authentication.
client 192.168.1.10 {
       # Shared secret between access point/controller and RADIUS server.
       secret = secret
       # Short name for the unit (for the RADIUS log)
       shortname = Cisco AP1131AG
}
client 192.168.1.11 {
       secret = secret
       shortname = Cisco AP1230
}
client 192.168.1.12 {
       secret = secret
       shortname = Cisco AP1100
}
```

proxy.conf

```
# How shall users defined as LOCAL be handled?
realm LOCAL {
        type            = radius
        authhost        = LOCAL
        accthost        = LOCAL
}
# How shall users with no realm be handled?
realm NULL {
        type            = radius
        authhost        = LOCAL
        accthost        = LOCAL
}
# How other defined realms shall be handled
realm uninett.no {
        type            = radius
        authhost        = LOCAL
        accthost        = LOCAL
}
# All other realms which are not defined
# Used in a RADIUS hierarchy to find the correct RADIUS server
realm DEFAULT {
        type            = radius
        authhost        = 158.38.0.237:1812
        accthost        = 158.38.0.237:1813
        secret          = supersecret
        nostrip
}
```

users

```
# Examples only. This file may be left without content.

# User “anonymous” may be denied access (NB: Everything must be on the same line!)
DEFAULT            User-Name  =~  "^[Aa][Nn][Oo][Nn][Yy][Mm][Oo][Uu][Ss]$",  AuthType:=
Reject

# This is how to deny access to users with no realm
DEFAULT         Realm == NULL, AuthType := Reject

# Used for testing and, if necessary, rebooting of freeRADIUS
eduroam-test        Auth-Type := Reject
                    Reply-Message = "samson3.uninett.no: OK"

# One may enter users with passwords in plain text or encrypted.
# Using the “users” file as a user database is not recommended.
```

6 Configuring an access point or controller

Access points and controllers are the units which perform the least complex part of the authentication process. Exactly how this is set up will of course vary from product to product, but in general the procedure is:

1. Create an SSID
2. Associate the SSID with a VLAN
3. Define the authentication mechanism to make use of 802.1X with EAP, or if required select WEP/WEP2
4. Define the encryption algorithm as TKIP and/or AES
5. Define RADIUS servers

On a Cisco access point this will appear as follows (only relevant entries are included):

```
aaa group server radius rad_eap
 server 158.38.0.237 auth-port 1812 acct-port 1813
!
aaa group server radius rad_acct
 server 158.38.0.237 auth-port 1812 acct-port 1813
!
dot11 ssid mynetwork
authentication open eap eap_methods
authentication key-management wpa
accounting acct_methods
!
interface Dot11Radio0
 encryption mode ciphers tkip
!
ssid eduroam
!
!
radius-server host 158.38.0.237 auth-port 1812 acct-port 1813 key secret
radius-server vsa send accounting
```

7 Configuring a client

There are as many ways of configuring as there are different clients. How they shall be configured will also depend on which options one has chosen for authentication and encryption.

In general, we can say:

1. If necessary, install the CA's public certificate.
2. Define a profile for the desired WLAN.
3. Specify the SSID.
4. Specify network authentication as WPA and/or WPA2.
5. Specify data encryption as TKIP and/or AES.
6. Specify EAP type as using either certificate (TLS) or user name/password (PEAP or TTLS).
7. Specify that you want to verify the server certificate and specify CA. Where possible you should also specify the name of the server so that this can also be verified.
8. For PEAP/TTLS: Specify authentication by MS-CHAPv2 and enter a user name and password if desired.

A few clients support the export and import of profiles, which makes this process simpler when there are several similar clients.

8 Dynamic assignment of VLAN

Using a RADIUS server makes it possible to send a certain amount of information back to the access point along with the approval of the user's authentication, the Attribute Value Pair (AV Pair). This is a very useful function which allows one to send instructions as into which VLAN the user shall be placed. This can also be done dynamically, so that one is assigned a VLAN dynamically according to the rights one has as a user, or if necessary according to the time of day. As long as the access point has the VLAN accessible in its trunk, the client will be placed there.

There are a number of ways of identifying a user in a user group. This may for example be entered as additional information from an LDAP query or may be evaluated on the basis of a prefix or suffix (realm) in the user name.

In FreeRADIUS such an AV Pair configuration may appear in a user's file as follows:

```
# Employees will be placed on VLAN 10
DEFAULT Realm == employee.uninett.no
 Service-Type = Login-User,
 Tunnel-Type  = VLAN,
 Tunnel-Medium-Type  = IEEE-802,
 Tunnel-Private-Group-Id  = 10,
 Fall-Through = yes

# Students will be placed on VLAN 20
DEFAULT Realm == student.uninett.no
 Service-Type = Login-User,
 Tunnel-Type  = VLAN,
 Tunnel-Medium-Type  = IEEE-802,
 Tunnel-Private-Group-Id  = 20,
 Fall-Through = yes
```

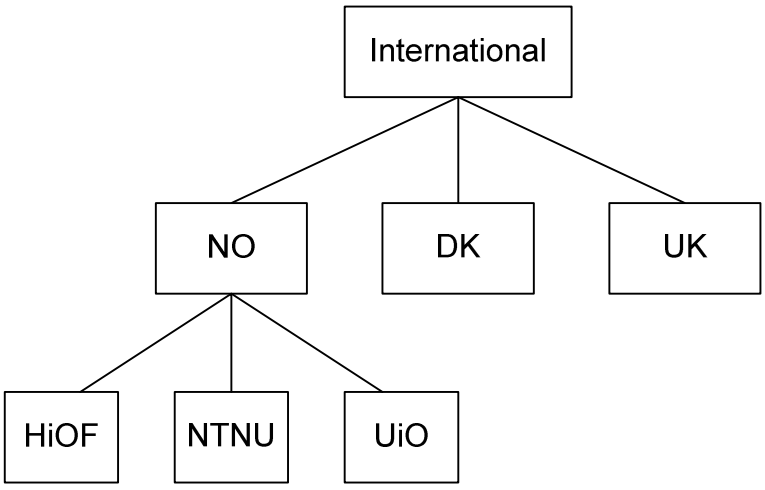
A user who achieves a successful login as "*user@employee.uninett.no*" will end up on VLAN 10 while a user who achieves a successful login as "*user@student.uninett.no*" (a completely different user) will end up on VLAN 20. The standard VLAN for a given SSID may very well be a third VLAN.

9 Eduroam

Eduroam is the result of a joint-venture project in TERENA (TF-Mobility) of which UNINETT is a member. Briefly, all users of eduroam will easily be able to log on to the wireless network of any institution in the world which is affiliated with eduroam.

This has been achieved by means of a system based on trust in which all members have connected their RADIUS servers to form a RADIUS hierarchy. There are a couple of international eduroam-dedicated top-level RADIUS servers to which all member countries have connected their respective national top-level RADIUS servers. In each country, all the eduroam-affiliated institutions have connected their RADIUS servers to the national RADIUS servers.

In Norway, UNINETT is responsible for the national RADIUS servers.



A necessary requirement is that the affiliated institutions shall offer a wireless network using the SSID “eduroam”. Moreover, the wireless network shall be protected using 802.1X authentication and at least TKIP encryption. The minimum number of ports which shall be open to the outside world is also specified.

In this RADIUS hierarchy, the RADIUS server of the organisation to which the user belongs will always perform the authentication. An authentication query is routed to the correct RADIUS server by using its realm, i.e. an organisation identification which follows the user name. For example, a user in UNINETT will be *user@uninett.no* while a user in the University of Oslo will be *user@uio.no*. The local RADIUS server will trust the result of the foreign authentication process when it grants access to guests.

For practical reasons, most institutions choose to use the SSID “eduroam” as the only SSID on their campus and to place users in the desired VLAN by means of dynamic VLAN assignment. This solution is chosen because it leads to less confusion regarding which SSID one should use, and provides the user with a well-known name and profile when visiting other institutions. Another important reason is that the user’s WLAN profile and configuration can be used unchanged anywhere where eduroam is available. Hence users do not need to make any changes in configuration, but connect to the foreign wireless network in exactly the same way as at home in their own institutions.

If one’s own wireless network is equipped with 802.1X and RADIUS, the transition to eduroam is very simple. In practice this involves setting up a special IP subnet for guests, the SSID “eduroam” for this subnet and a little additional configuration of the RADIUS server so that authentication from an unknown realm is forwarded to the national top-level RADIUS servers.

Additional information about eduroam and the Policy Document for access is found at http://www.eduroam.no.

10 Summary and recommendation

Based on the methods described in this document, we here present a summary of our recommended security mechanism for wireless networks.

Authentication:
- Use IEEE 802.1X

Authentication protocol:
- PEAP
- TTLS
- TLS, if one has PKI for this.

These protocols are not mutually exclusive and can be supported simultaneously.

RADIUS server:
- FreeRADIUS
- Microsoft IAS/NPS
- Cerebrum

One is free to choose a RADIUS server according to one's preferences but not all servers support the EAP methods PEAP, TTLS and TLS, which should be a minimum requirement (the above-mentioned support these).

Encryption:
- At least TKIP (WPA)
- Preferably AES (WPA2)

It would be best to use only AES but for reasons of backward compatibility the majority of networks must also support TKIP. Both can be available simultaneously, but some clients may experience problems obtaining access. The choice will depend on local conditions.

Certificate:
- Self-generated
- Purchased from a known CA.

The choice is optional and will depend on one's own situation. Theoretically, a self-generated certificate is more secure but in practice the risk associated with a purchased certificate is low. One must have a certificate for one's RADIUS server but whether or not to use it also for users will be based on whether one wishes to use a user database or manage a PKI.

User rights
Allocate users to different VLANs depending on their user groups – employees, management, technical staff, students, and so on. Use RADIUS with dynamic VLAN assignment and only one SSID. The desired VLAN may be assigned by the RADIUS server or be obtained from the user database if this contains the necessary information. Use ordinary network filters to assign rights to the different VLANs.

User database
For many, the user database will ideally be a previously established user database, making it unnecessary to create a new one only for wireless authentication. Many different types of user databases exist. The choice

should be one which is easy to manage, which saves passwords in a secure way and which makes passwords available for verification via MS-CHAPv2, i.e. in an MD4 format. It must of course also be capable of communicating with the type of RADIUS server that is being used.

Eduroam
When one has IEEE802.1X and a RADIUS server, very little technical configuration will be necessary to access eduroam. One requirement is that the SSID shall be “eduroam”. The advantage is the ability to access the wireless networks of every other eduroam institution worldwide.

Simple, autonomous APs (SoHo products)
Use WPA-PSK with at least 20 characters in the password. Keep to alphanumerical characters since the key generation routines of certain products do not function correctly with special characters.

Special remarks regarding autonomous APs

- Do not let the access point’s administrative IP address be accessible to users. Place the IP addresses on an administrative network which is not accessible wirelessly.
- In the case of several VLANs, the AP must have a trunk. Remember to safeguard the cable against access by unauthorised persons who may thereby gain access to an otherwise protected VLAN. Do not let the trunk contain more VLANs than necessary for the wireless network.
- Use encrypted communication for configuring (SSH/HTTPS).

Special remarks regarding wireless infrastructure with controllers

- Assign access points to a separate administrative network which is only permitted to communicate with the controller (all traffic in and out goes through the controller).
- Use encrypted communication for configuring (SSH/HTTPS).
- Take advantage of the controller’s functions and actively monitor whether suspicious access points appear in your network.

References

eduroam – An academic RADIUS hierarchy which gives users access to other WLANs.
http://www.eduroam.no/

FreeRADIUS – A RADIUS server based on open source code.
http://www.freeradius.org/

IEEE – The European standardisation organization.
http://www.ieee.org/
http://standards.ieee.org/getieee802/802.11.html

TERENA – A European collaborative organisation for European National Research and Education Networks (NRENs).
http://www.terena.org/

Wi-Fi Alliance – Certification of compatible wireless products.
http://www.wifialliance.com/

Glossary

AES: Advanced Encryption Standard, a very secure encryption algorithm.

CA: A Certificate Authority is an entity that issues certificates which other parties can use. A CA is a reliable third party which all parties should be able to use to verify each other's credentials.

BSSID: Basic Service Set ID, the MAC address of a wireless access point.

EAP: Extensible Authentication Protocol, a protocol which conveys the authorisation protocol in an 802.1X secured network.

EAPOL: EAP over LAN, the EAP protocol which runs on Layer 2 of a network.

IEEE: Institute of Electrical and Electronics Engineers, an organisation which prepares standards for data communication, among other things.

IEEE 802.11: The first standard for wireless networks as we are familiar with them today. It was based on the use of frequencies in the 2.4 GHz band but defined two different transmission technologies: Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS). Both had data transfer rates of 1 Mbps and 2 Mbps. The use of IR was also catered for but was never implemented for the market. IEEE 802.11 DSSS formed the basis of IEEE 802.11b. DSSS defined channels 1 to 11 (USA), 1 to 13 (Europe) and 1 to 14 (Japan) for the 2.4 GHz waveband. Some of these channels overlap.

IEEE 802.11i: A standard for the use of 802.1X and TKIP or AES encryption in wireless networks.

IEEE 802.1X: A standard for port-based authentication using the EAP protocol. A port may be a physical network contact in a switch or a virtual connection between an associated client and an access point.

MAC: Media Access Control, a unique address which all network units must have.

PKI: Public Key Infrastructure, a hierarchical system in which a user has a CA and associated server, and user certificates. Each certificate is unique and can be verified by the CA.

RADIUS: Name of the protocol used by a RADIUS server. A RADIUS server is an authentication server.

SAMSON 3: UNINETT's standard Linux server for e-mail, DNS, web server, RADIUS and other applications

SCS: Server Certificate Service

SSID: Service Set ID, identifies a specific wireless network. Several access points may have the same SSID in order to access a larger network.

SSL: Secure Sockets Layer, an encrypted communication channel.

TKIP: Temporal Key Integrity Protocol, an RC4 encryption which is an improvement of WEP.

Wireless network: The term "wireless network" is basically not very specific and may apply to many different types of technology. This document defines wireless networks as specified by the standards IEEE 802.11, IEEE 802.11b, IEEE 802.11g, IEEE 802.11a and IEEE 802.11n.

VPN: Virtual Private Network, an encrypted communication channel.

WEP: Wired Equivalent Privacy, wireless encryption defined by IEEE 802.11.

Wi-Fi Alliance: A special interest group of which almost all manufacturers of wireless equipment are members. It prepares test criteria and certifies products which satisfy these criteria. The intention is to provide security for consumers who should be able to rely on a product they buy to work in combination with other suppliers' products.

WPA: Wi-Fi Protected Access. Certification of equipment which satisfies the requirements of the Wi-Fi Alliance with regard to functionality in accordance with the IEEE 802.11i standard (i.e. support for TKIP encryption).

WPA2: Wi-Fi Protected Access 2. Certification of equipment which satisfies the requirements of the Wi-Fi Alliance with regard to functionality in accordance with the IEEE 802.11i standard (i.e. support for AES encryption).