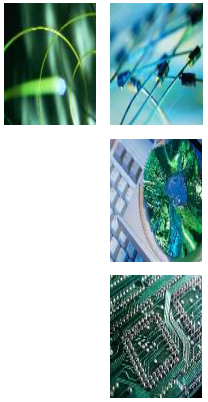
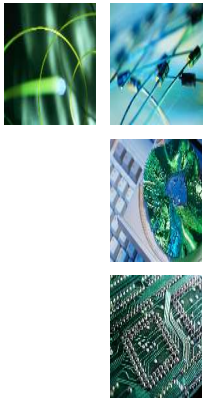


IEEE 802.1X workshop

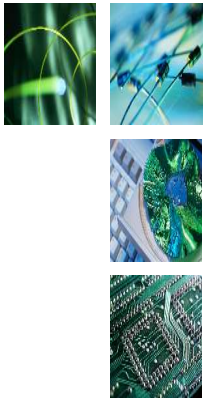
Networkshop 34, 4 April 2006.

Josh Howlett, JRS Technical Support, University of Bristol.



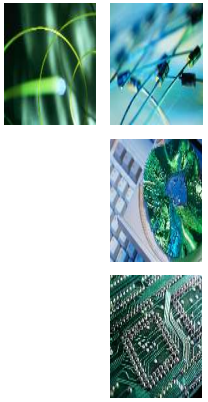
Introduction

- **Introduction** (5 mins)
- **Authentication overview** (30 mins)
- **IEEE 802.1x** (20 mins)
- **Securing 802.11 with .1x** (20 mins)
- **Break** (15 mins)
- **Campus deployment** (20 mins)
- **802.11 key management** (10 mins)



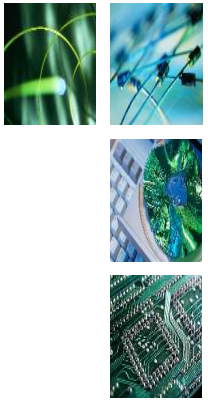
Introduction

- Objectives
  - Provide an understanding of:
    - network authentication in general;
    - how 802.1x works at a technical level;
    - how 802.1x can be deployed to address network security and management issues.
  - Provide practical demonstrations of:
    - installing, configuring and testing a basic 802.1x environment;
    - debugging 802.1x problems.



Authentication overview

- Ancient history
  - In the beginning was the serial line...
  - **IP over Serial Line** (SLIP)
    - *de facto* standard;
    - very basic protocol: no peer negotiation of the connection whatsoever (network parameters, authentication, etc);
    - fine for small numbers of connections over trusted links, but otherwise unmanageable.

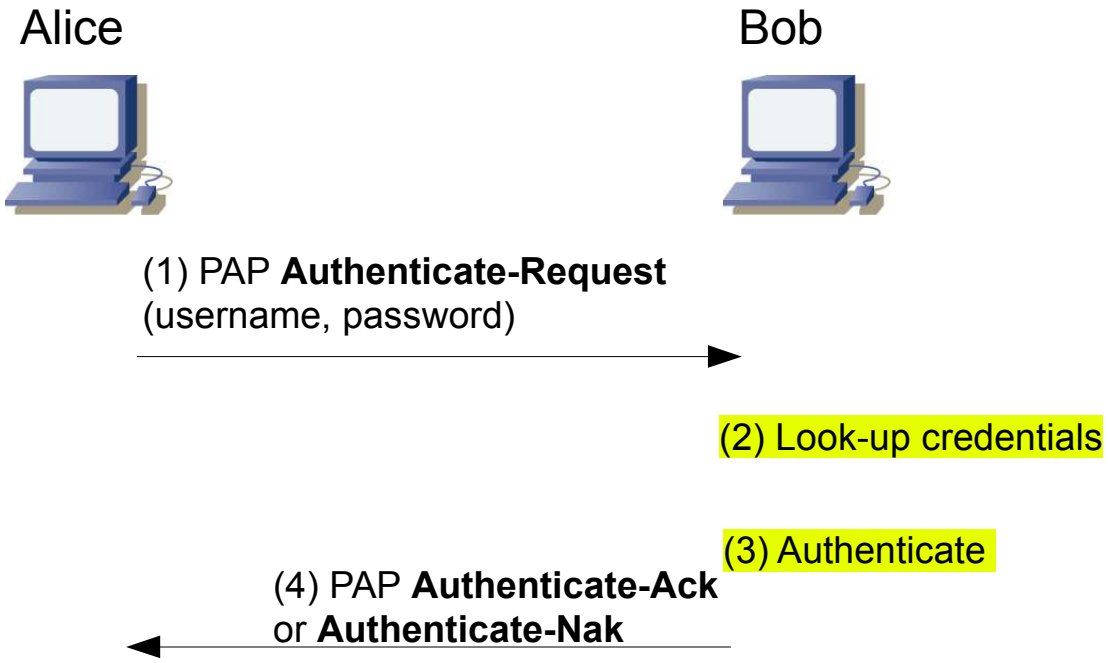


Authentication overview

- Point to Point Protocol (**PPP**)
  - Provides a means for PPP **peers** to negotiate an authentication protocol:
    - **Password Authentication Protocol** (**PAP**)
      - Alice sends user-name and password to Bob.
    - **Challenge Authentication Protocol** (**CHAP**)
      - Alice sends Bob a random number (**Challenge**).
      - Bob hashes (MD5) his password with the challenge, and returns it to Alice (**Response**).
      - Alice hashes her copy of Bob's password. If the hashes match, Bob has the right password.

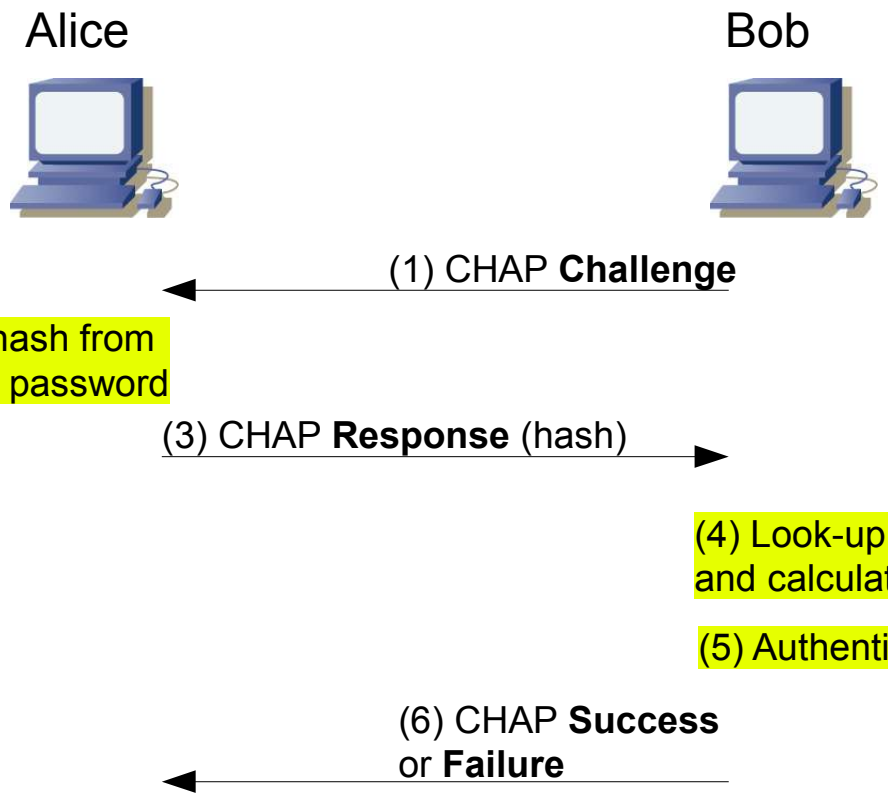
Authentication overview

PAP authentication



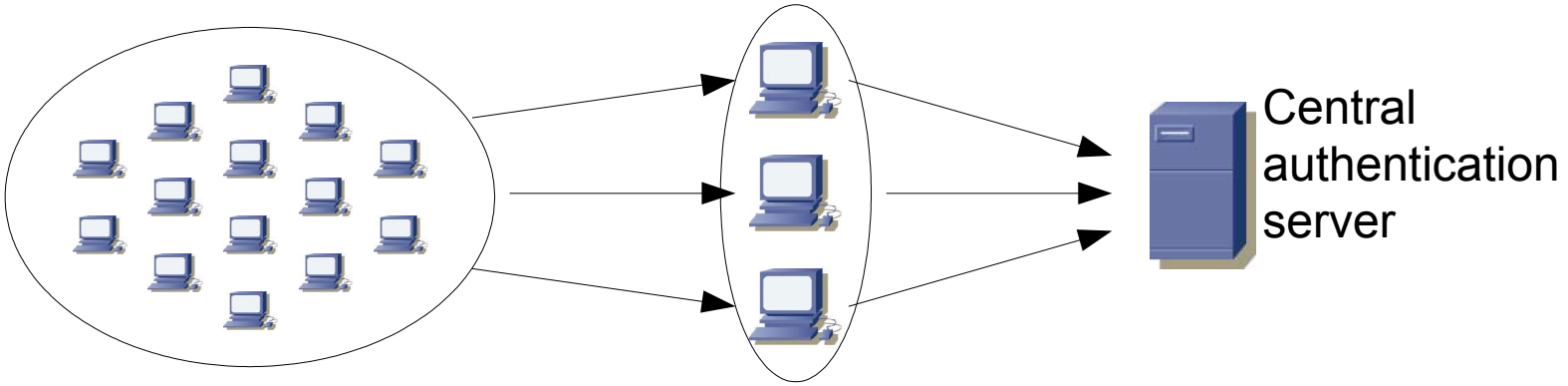
Authentication overview

CHAP authentication

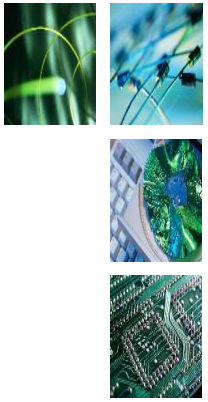


Authentication overview

- Problems with PPP authentication (1)
  - Scaling authentication
    - If the number of possible PPP peers that requires authentication becomes large, it is desirable to centralise authentication.



Authentication overview



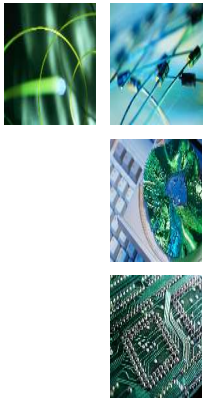
- Scaling authentication with RADIUS
  - **Remote Access Dial-up User Service** (**RADIUS**)
    - The peer providing network connectivity is called the **Network Access Server** (**NAS**).
    - The NAS does not attempt to authenticate peer's credentials itself.
    - The NAS acts as a **RADIUS client**, sending the credentials to a **RADIUS server**.
    - The NAS enforces the decision (**Accept**, **Reject**) returned by the RADIUS server.

Authentication overview

PAP authentication with RADIUS

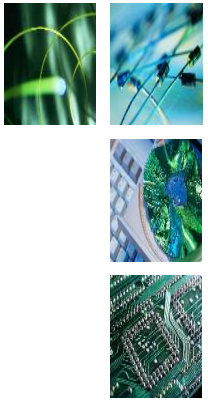
Authentication overview

CHAP authentication with RADIUS



Authentication overview

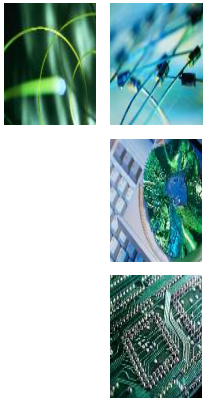
- Microsoft extensions to CHAP
  - Windows stores user credentials as **LM** and **NT** hashes, making CHAP impossible.
  - MS-CHAP-v1
    - Uses either the LM or NT hash to calculate the Response, and not the clear-text password.
  - MS-CHAP-v2
    - Removes LM, and adds **mutual authentication** by sending a challenge to the NAS in the CHAP Response. The NAS' response is included in a CHAP Success packet.



Authentication overview

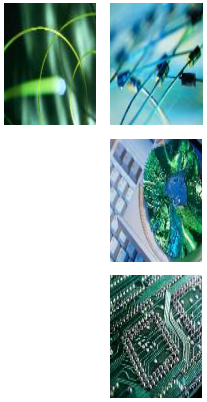
- FreeRADIUS
  - Configure and test PAP and CHAP
    - /etc/raddb/users
      - » `testuser  User-Password = "testpass"`
    - Invoke FreeRADIUS in debug mode
      - » `radiusd -X`
    - Testing PAP with radtest
      - » `echo "User-Name = testuser, CHAP-Password = testpass" | radclient localhost auth testing123 -x`
    - Testing CHAP with radclient
      - » `echo "User-Name = testuser, User-Password = testpass" | radclient localhost auth testing123 -x`

Authentication overview



- Problems with PPP authentication (2)
  - The PPP authentication model
    - PAP: not very secure!
    - CHAP: needs access to plain-text password.
    - More generally
      - PPP peers must share at least one mutually acceptable authentication protocol. This makes it hard to quickly deploy better authentication protocols in heterogeneous environments.
      - A PPP peer must “understand” the authentication protocol; even if only to pack the client's credentials into a RADIUS packet in the right format.

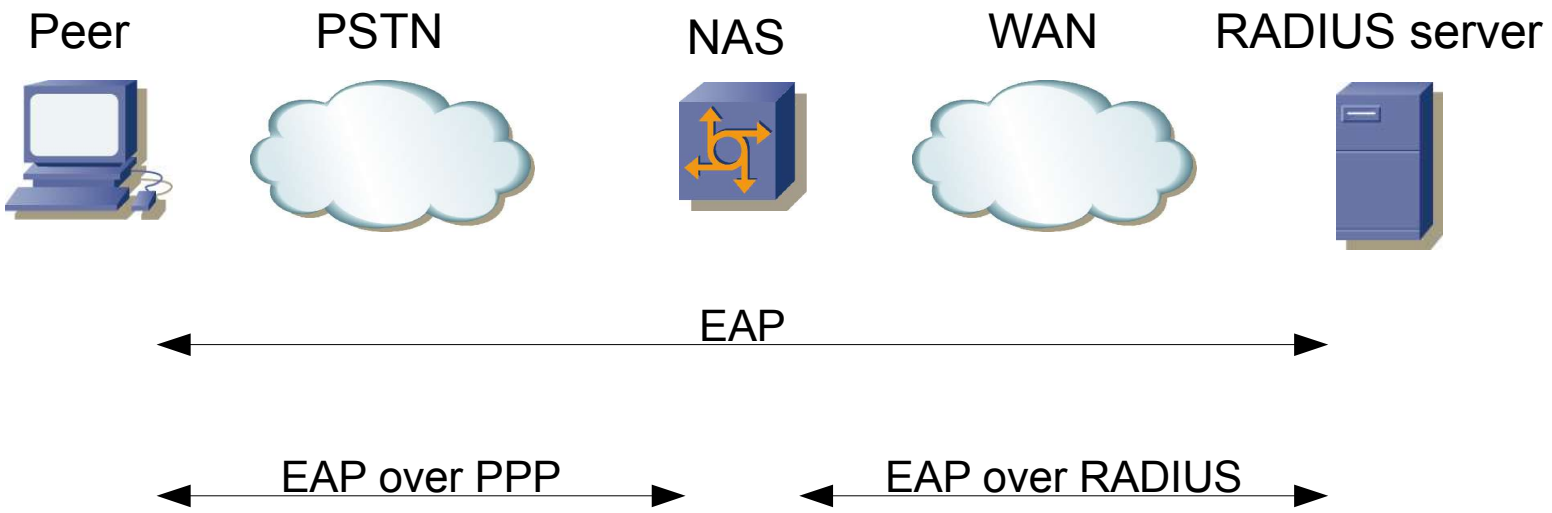
Authentication overview

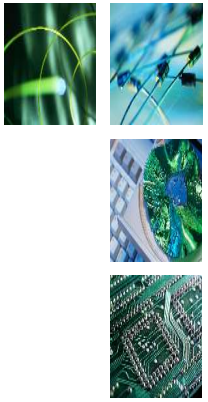


- **Extensible Authentication Protocol (EAP)**
  - EAP cuts out the middle man (the NAS).
  - This permits a direct conversation between the peer being authenticated and the RADIUS server; the NAS treats the EAP conversation as a “black box”.
  - New EAP-based authentication protocols (“EAP types”) can therefore be deployed more easily.

Authentication overview

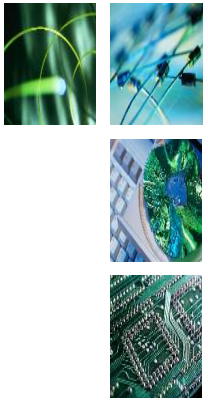
EAP over LAN & EAP over RADIUS





Authentication overview

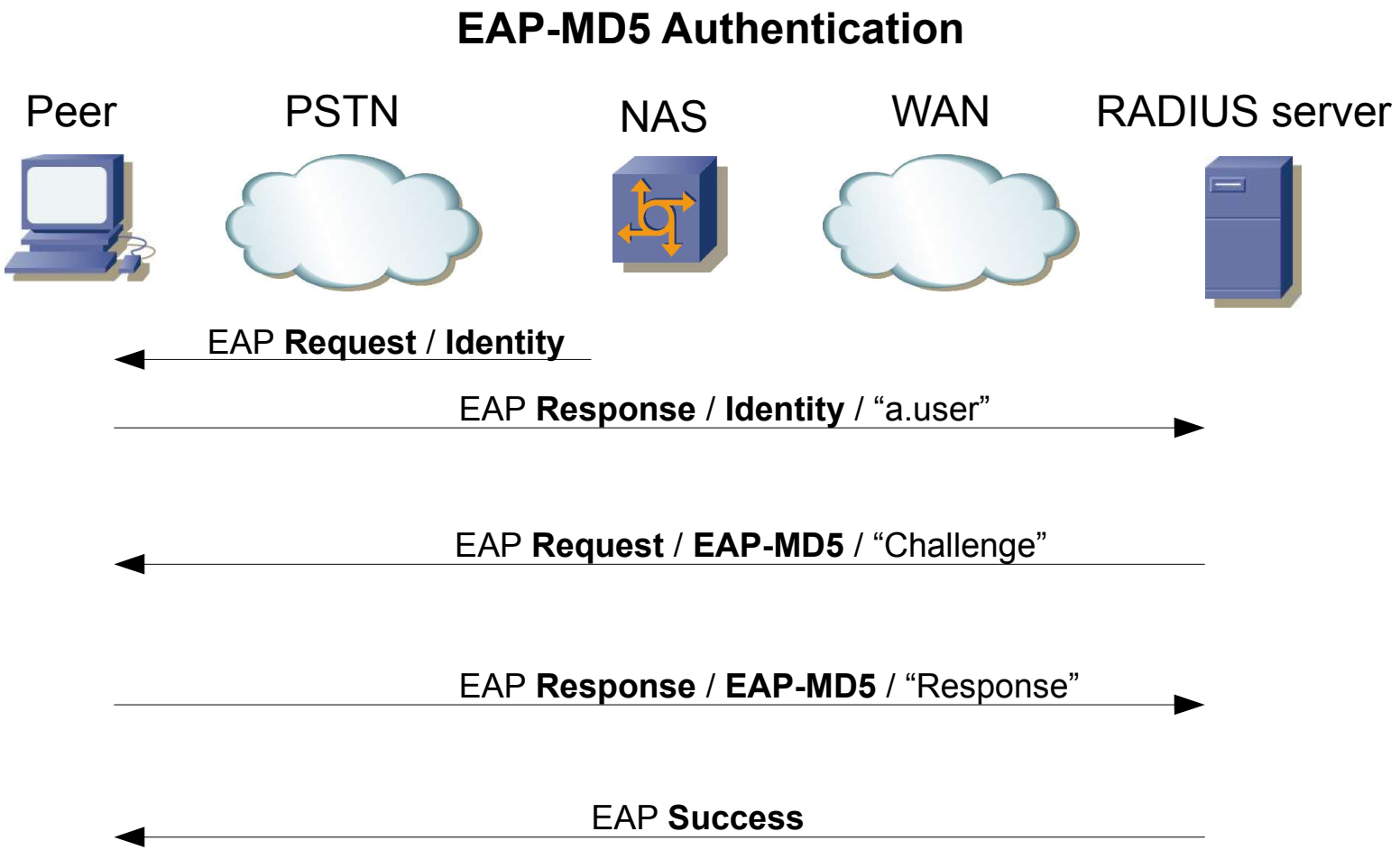
- EAP packet format
  - All EAP packets have the following fields
    - Code
      - **Request**, **Response**, **Success**, **Failure**.
    - Identifier
      - Used to match Requests with Responses.
    - Length
      - Length of the entire packet
    - Data



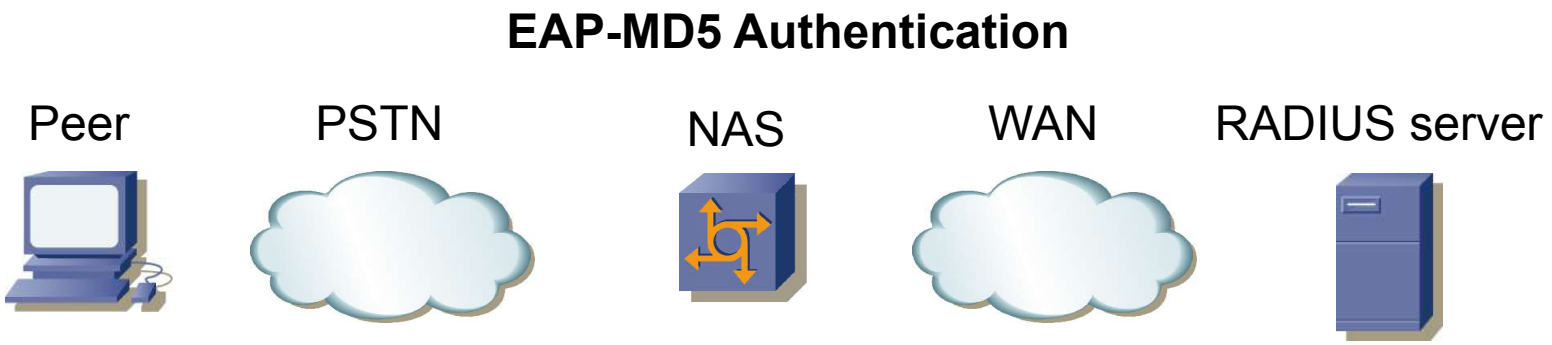
Authentication overview

- EAP types
  - Request and Response packets only
    - Non-authentication types
      - **EAP-Identity**: used to request the peer's identity.
      - **EAP-Notification**: used to send a notification.
      - **EAP-Nak**: used to indicate a proposed type is unacceptable
    - Authentication types
      - **EAP-MD5**: analogous to CHAP.
      - **EAP-OTP**: RFC 2289 OTP authentication.
      - **EAP-GTC**: Supports other OTP systems.

Authentication overview

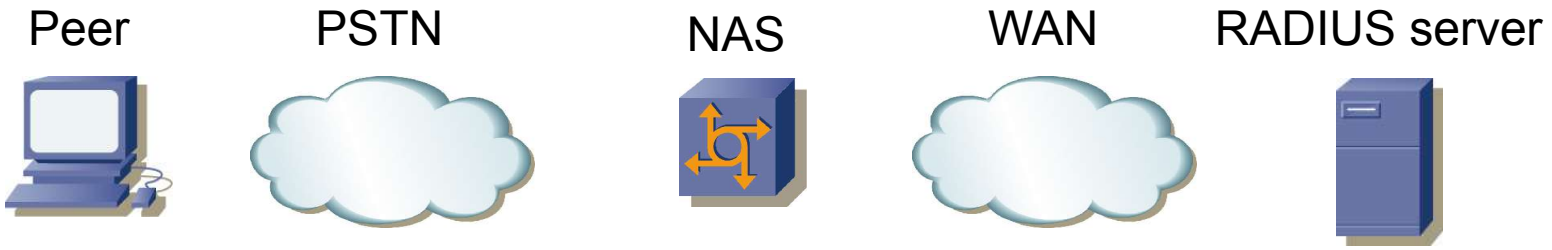


Authentication overview



Authentication overview

Example of EAP Nak negotiation



EAP **Request** / **Identity**

EAP **Response** / **EAP-Identity** / “a.user”

EAP **Request** / **EAP-OTP** / “Challenge”

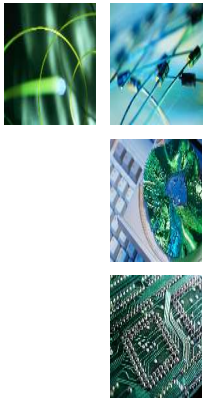
EAP **Response** / **EAP-Nak** / “EAP-MD5”

EAP **Request** / **EAP-MD5** / “Challenge”

EAP **Response** / **EAP-MD5** / “Response”

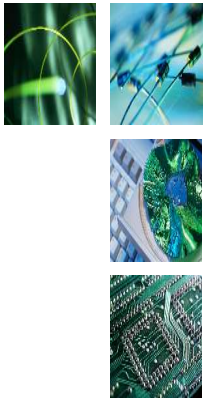
EAP **Success**

Authentication overview



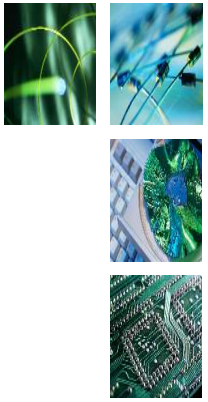
- FreeRADIUS EAP-MD5
  - No FR configuration changes required
    - `eapol_test -c supplicant.cfg -a localhost -s testing123`
      - supplicant.cfg:

```
Network={
    eap=MD5
    identity="testuser"
    password="testpass"
    key_mgmt=IEEE8021X
}
```



IEEE 802.1x

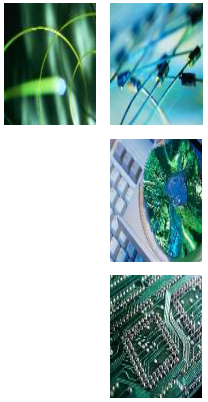
- Where does 802.1x fit in?
  - Pressures on the network
    - rapid growth in host numbers;
    - increasing mobility of hosts;
    - flood-wiring;
    - wireless.
  - Lead to problems with
    - security;
    - management.



IEEE 802.1x

- What is 802.1x?
  - "Port-based network access control".
  - Operates at the link layer (ethernet):
    - **Supplicant**: peer requesting authentication
      - Typically an end-user host.
    - **Authenticator**: peer demanding authentication
      - Typically a switch or access point.
  - The authenticator does not forward data until supplicant the is authenticated.

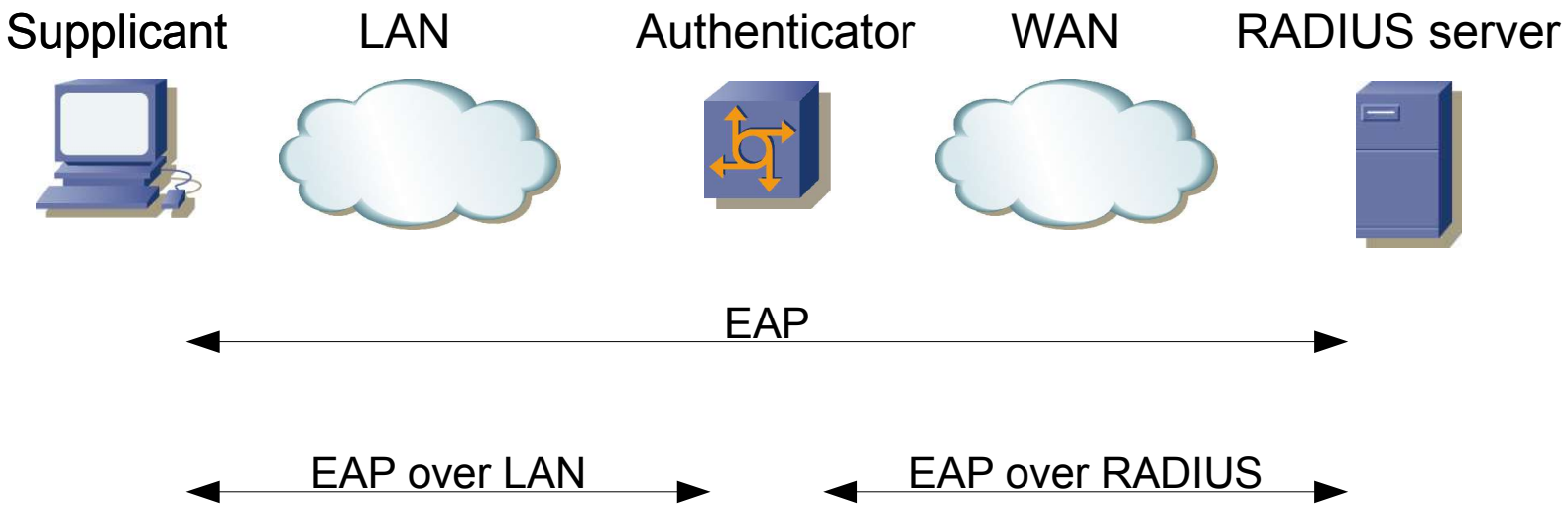
IEEE 802.1x

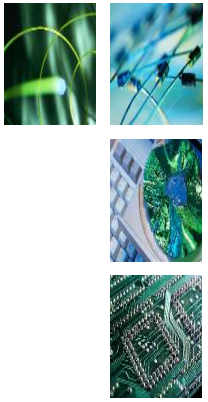


- What is 802.1x?
  - EAP is used for 802.1x authentication.
  - Two EAP encapsulations
    - **EAP over LAN** (**EAPOL**):
      - Between the supplicant and the authenticator.
      - The EAP packet is encapsulated directly in an ethernet frame.
    - EAP over RADIUS:
      - Between the authenticator and the RADIUS server.
      - The EAP packet is encapsulated within a RADIUS attribute ("**EAP-Message**").

IEEE 802.1x

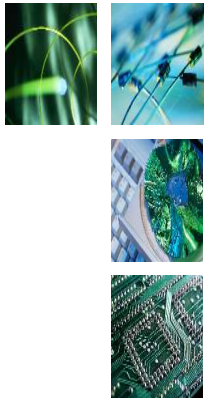
EAP and 802.1X





IEEE 802.1x

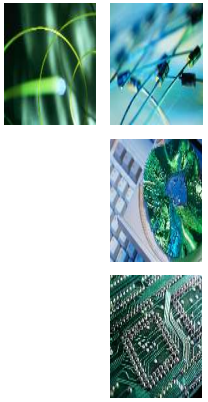
- Communication channels
  - EAP
    - Between supplicant and RADIUS server.
    - Generally only used for authentication.
    - Projected uses include **Network Admission Control** and **MobileIP**.
  - RADIUS
    - Between authenticator and RADIUS server.
    - Used for many purposes.
    - A popular use is **dynamic VLAN allocation**.



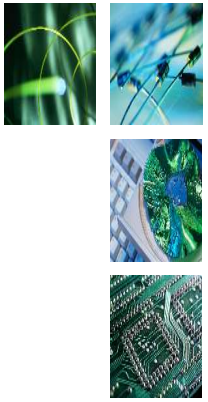
IEEE 802.1x

- Configuring dynamic VLAN allocation
  - /etc/raddb/users
    - testuser User-Password=”testpass”
      Tunnel-Type := "VLAN",
      Tunnel-Medium-Type := "IEEE-802",
      Tunnel-Private-Group-Id := "VLAN0012"

Securing 802.11 with .1x

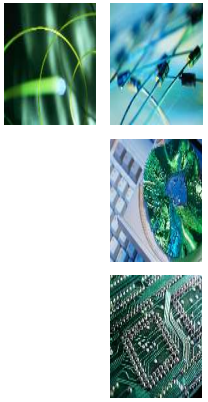


- 802.11 security
  - Originally required **static keys**.
  - Clients / WAPs could have 1 to 4 keys.
  - Simple, but useless in practise:
    - Manual distribution of key(s) to clients;
    - Must trust end-users to look after their key(s), and not to snoop on the network;
    - The compromise of a single key requires re-keying those clients that also use that key.
  - 802.11 security doomed?



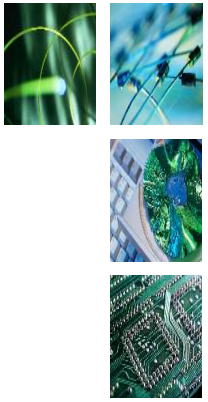
Securing 802.11 with .1x

- EAP to the rescue!
  - Obscure EAP method called **EAP-TLS**
    - Original application was for PPP connections
      - "PPP EAP TLS Authentication Protocol"
    - Uses **Transport Layer Security** (**TLS**)
      - authentication using **public key cryptography**;
      - enables derivation of a **secret encryption key**, and negotiation of cryptographic algorithms;
      - encryption of application data using the secret key and the negotiated algorithms.



Securing 802.11 with .1x

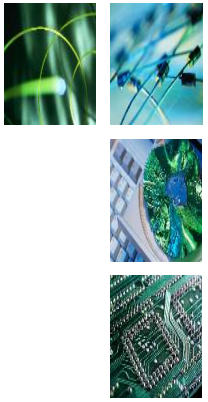
- TLS
  - TLS **Handshake** protocol
    - Messages
      - Client Hello (acceptable algorithms);
      - Server Hello (server certificate, acceptable algorithms, request client certificate);
      - Client Finished (client certificate);
      - Server Finished.
    - Result: agreement of a key and algorithms.
  - TLS **Record** protocol
    - Key and algorithms protect application data.



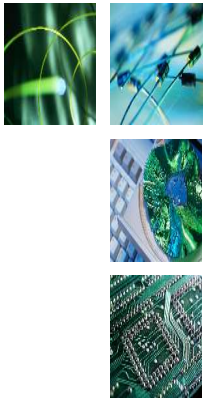
Securing 802.11 with .1x

- **EAP-TLS**
  - TLS handshake over EAP
    - The TLS Record protocol is not used.
  - Advantages
    - Strong authentication;
    - Derivation of keys for encryption.
  - Disadvantages
    - User or machine certificates required!

Securing 802.11 with .1x



- **EAP-PEAP**
  - Protected EAP; takes TLS a step further...
  - A two phase protocol
    - Phase 1: TLS handshake
      - as EAP-TLS, but without a client certificate.
    - Phase 2: TLS Record exchange
      - protects a second "**inner**" EAP method used to authenticate the client.
    - The inner method is typically EAP-MS-CHAPv2.

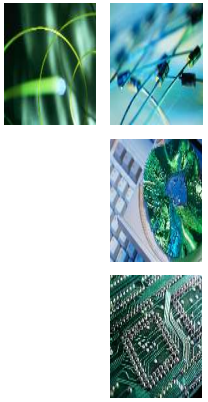


Securing 802.11 with .1x

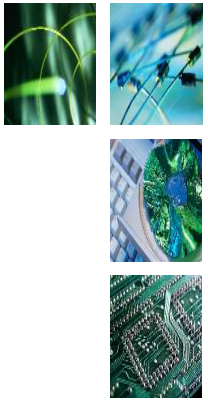
- Configuring EAP-PEAP
  - /etc/raddb/eap.conf
    - uncomment the TLS and PEAP sections.
  - eapol_test command same as EAP-MD5.
    - supplicant.cfg

```
» Network={
    eap=PEAP
    identity=”testuser”
    password=”testpass”
    key_mgmt=IEEE8021X
  }
```

Securing 802.11 with .1x



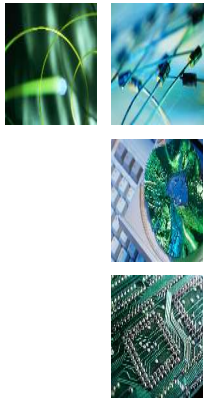
- **EAP-TTLS**
  - Tunnelled TLS; similar to PEAP...
  - A two phase protocol again
    - Phase 1: TLS handshake
      - as EAP-TLS, but without a client certificate.
    - Phase 2: TLS Record exchange
      - protects a second “inner” RADIUS-based exchange.
    - Because the inner protocol is RADIUS, TTLS can use virtually any authentication protocol.



Securing 802.11 with .1x

- Configuring **EAP-TTLS**
  - /etc/raddb/eap.conf
    - uncomment the “TLS” and “TTLS” sections.
  - eapol_test command same as EAP-MD5.
    - supplicant.cfg

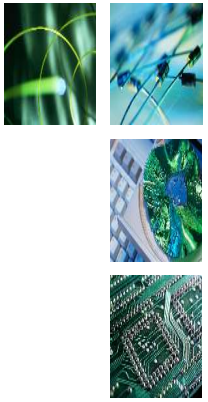
```
» Network={
    eap=TTLS
    identity=”testuser”
    password=”testpass”
    key_mgmt=IEEE8021X
  }
```



Securing 802.11 with .1x

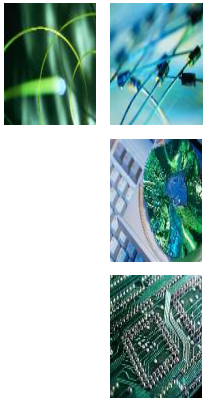
- **Tunnelled** EAP considerations
  - The client needs the RADIUS server's CA's **root certificate**.
  - The CA may be **self-signed**
    - if it is self-signed, the root certificate will need to be distributed to clients.
    - if it is not self-signed, care must be taken to configure supplicants to verify the CN.
  - Outer EAP method "**Identity hiding**".
  - The **cryptographic binding** problem.

Campus deployment

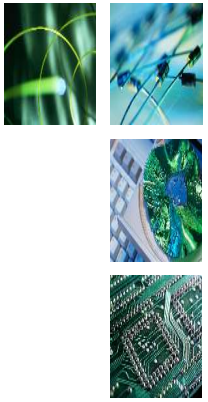


- Campus deployment considerations
  - Access hardware
  - Supplicant software for workstations
  - Authentication back-end

Campus deployment



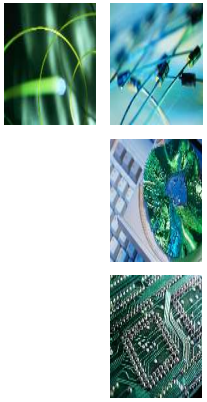
- Access hardware
  - Most modern kit supports 802.1x and is trivial to configure.
  - Cisco access points have some gotchas:
    - 1-to-1 **SSID** to VLAN binding;
    - if using dynamic VLAN allocation, cipher on both VLANs must be the same;
    - no dynamic VLAN allocation with multiple broadcast SSIDs;
    - limited to 16 VLANs, and no **VTP**;



Campus deployment

- Supplicant software
  - Microsoft supplicant (Windows XP SP2)
    - EAP-MD5 (wired only), EAP-TLS, EAP-PEAP.
    - Three authentication modes
      - **user** and/or **machine** authentication.
    - Problems
      - caching of credentials;
      - no group policies on wired interfaces;
      - some registry hacks tweaks are desirable;
      - the same EAP type must be used for user and machine authentication if both are required.

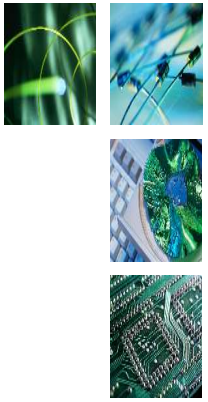
Campus deployment



- Supplicant software
  - Other supplicants
    - From UKERNA's “802.1X” Technical Sheet

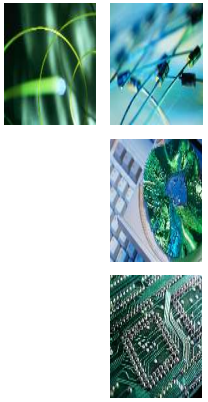
	Operating systems								EAP types			802.11 ciphers			Other	
Supplicants	***W95***	***W98***	***WME***	***W2K***	***WXP***	***Linux***	***OSX***	***PPC***	***TLS***	***PEAP***	***TTLS***	WEP	***WPA***	***WPA2***	***Availability***	***Ease of use***
Native Windows				✔[1]	✔			✔		✔[2]		✔	✔[3]	✔[4]	Included with Windows XP	☺☺[5]
Native MacOS							✔[6]		✔	✔	✔	✔	✔	✔[7]	Included with MacOS X	☺☺☺
HP ProCurve Client		✔	✔	✔	✔	✔[8]	✔[9]		✔	✔	✔	✔	✔		Commercial	☺☺☺
Funk Odyssey		✔	✔	✔	✔			✔	✔	✔	✔	✔	✔	✔	Commercial	☺☺☺
Meetinghouse Aegis		✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔		Commercial	☺☺☺
SecureW2				✔[1]	✔			✔			✔	✔	✔[10]		Free (GPL licence)	☺☺☺
wpa_supplicant	✔[11]	✔[11]	✔[11]	✔	✔	✔			✔	✔	✔	✔	✔	✔	Free (GPL/BSD licences)	☺☺[12]
Xsupplicant						✔			✔	✔	✔	✔	✔	✔	Free (GPL/BSD licences)	☺☺☺
Wire1X		✔	✔	✔	✔				✔	✔	✔				Free (GPL/BSD licences)	☺[13]

Campus deployment



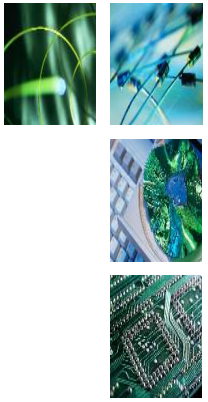
- Authentication back-end
  - Main considerations
    - the type of credentials;
    - the type of user database;
    - preferred vendor / technologies;
    - authorisation policy complexity.

Campus deployment



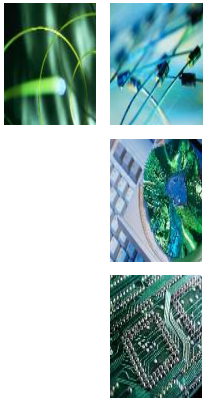
- Authentication back-end
  - The type of credentials determines the EAP types that can be used
    - clear text password (or reversibly encrypted)
      - Anything! (must be explicitly enabled in AD)
    - generic hash (DES, MD5, SHA, etc)
      - EAP-TTLS with PAP
    - NTLM hash (ie. AD, Samba)
      - EAP-PEAP; EAP-TTLS with PAP
    - X.509 user certificates
      - EAP-TLS; EAP-TTLS/EAP-PEAP with EAP-TLS

Campus deployment



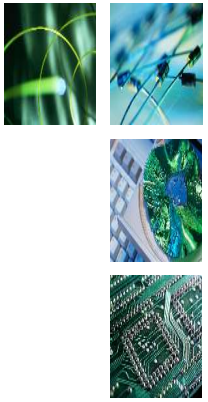
- Authentication back-end
  - The type of user database can influence the RADIUS server
    - AD: most RADIUS servers;
    - LDAP: FreeRADIUS, Radiator;
      - “authenticated bind” - EAP method must provide RADIUS server with bind-able credentials;
      - “query-and-compare” - RADIUS server may require privileged access to users' credentials.
    - FreeRADIUS & Radiator support a variety of other back-ends, and the ability to script.

Campus deployment



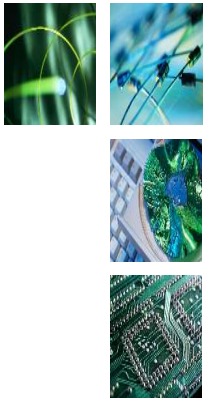
- Authentication back-end
  - Preferred vendor / technologies;
    - Microsoft
      - Internet Authentication Server against AD;
      - IAS does not support EAP-TTLS.
      - you may want to consider Radiator.
    - Novell
      - FreeRADIUS against NDS.
    - Unix / Linux
      - FreeRADIUS or Radiator against whatever!

Campus deployment



- Authentication back-end
  - Authorisation policy complexity
    - IAS
      - does not provide much scope for defining complex policies.
      - your mileage may vary, depending on your requirements.
    - FreeRADIUS and RADIATOR
      - possible to build very complex policies by scripting or building custom modules;
      - be prepared to get your hands dirty!

Campus deployment



- Planning

1) Wired, wireless or both?

2) If wireless, examine vendor documentation for gotchas!

3) What type of credentials, and where? (Active Directory? NDS? User certificates? SQL database?)

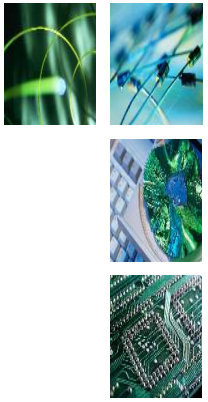
4) Given (1) and (3), determine the EAP types you could use.

5) Given (3) and (4) and other local factors (vendor preference, etc), select a suitable RADIUS server.

6) Machine and/or user authentication?

7) Given (4) and (6), select a suitable supplicant(s).

8) Resist the temptation to “build it in one go”. Build up a test system in small steps, adding new components as you go.

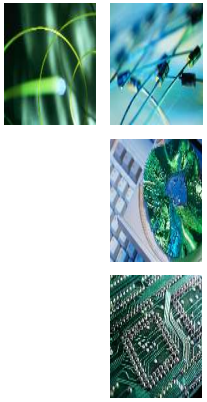


802.11 key management

- 802.11 key management
  - Most encryption ciphers use keys.
  - The client and access point must share the same key to talk to each other.
  - **Key management** is the most difficult aspect of cryptography.

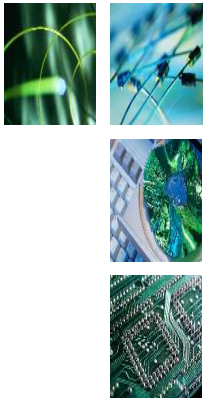
"Key management includes all of the provisions made in a crypto-system design, in cryptographic protocols in that design, in user procedures, and so on, which are related to generation, exchange, storage, safeguarding, use, vetting, and replacement of keys"

802.11 key management

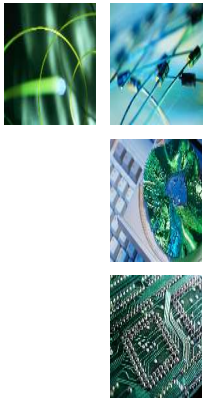


- Dynamic WEP key management
  - **MSK** generated by TLS exchange, and exported by EAP method to RADIUS server.
  - MSK placed in an encrypted RADIUS attribute, and sent to access-point.
  - Access-point generates a new random key, encrypts it with the MSK and sends it within an EAPOL frame to the client.
  - The new key is used for WEP encryption.

802.11 key management

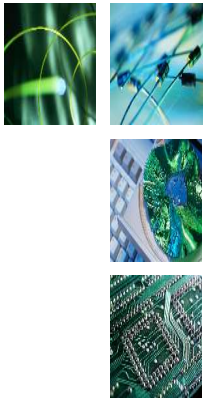


- Dynamic WEP key problems
  - Re-use of MSK in different contexts
    - TLS Record & encryption of WEP key.
    - Poor cryptographic practise.
  - No **re-keying**
    - Lack of re-key greatly increases the attack surface on the WEP key, which is already a serious problem given WEP's design problems.
    - Re-keying requires re-authentication.



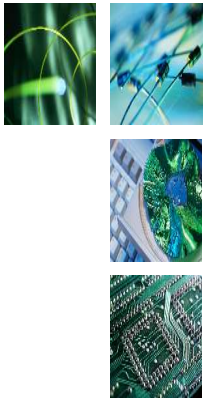
802.11 key management

- WPA key management fixes
  - **Pairwise Master Key** (**PMK**) from MSK
    - PMK derived from the MSK. The MSK never leaves the RADIUS server.
  - Temporal keys
    - PMK is used in combination with the **four-way handshake** to derive the **Pairwise Temporal Keys** (**PTK**).
    - The **per-packet encryption key** is derived from the **PTK**.



802.11 key management

- WPA2 key management enhanced
  - **Fast re-authentication**
    - PMK cached by client and APs.
    - Only a handshake is needed to authenticate.
  - **Pre-authentication**
    - Client picks up beacons from APs in range.
    - Client authenticates with these APs through the WAP it is currently associated with.
    - PMK cached on client and APs to allow fast re-authentication if they associate.



It's over!

Thank you for your attention!

Any questions?