

WLAN Security

EAP-FAST

(C) Herbert Haas 2006/4/1

NOTE: This material is for internal use only.

Content

In this chapter a detailed overview about today's WLAN security problems and solutions are presented.

This subchapter provides an introduction into EAP-FAST, which is considered as the successor of LEAP.

Objective

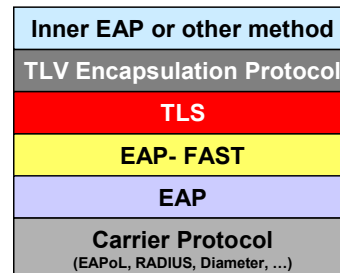
After completing this chapter the following tasks could be solved:

- Emphasize the basic vulnerabilities of WLAN
- Explain why WEP is insecure and give a mathematical example
- Compare WEP and TKIP/MIC
- Highlight the design flaws of the WLAN standard authentication
- Explain the design idea of 802.1x
- Compare EAP-TLS, LEAP, PEAP, EAP-TTLS, EAP-FAST with each other and emphasize important security features
- Explain the design concept of WPA and WPA2
- Implement a reliable 802.1x infrastructure over a WAN connection
- List important issues to be considered when choosing a VPN design
- Explain PSPF

Quick Facts



- **Cisco, LEAP successor**
 - ♦ Design by Cisco but open draft (IETF)
 - ♦ Initially known as "Tunneled EAP (TEAP)" or "LEAPv2"
 - ♦ Supported by client devices since Q4/2004
- **Goals:**
 - ♦ PEAP/EAP-TTLS -like security
 - ♦ Simple deployment
 - ♦ Fast roaming support (VoIP)
 - ♦ Computationally lightweight
 - Symmetric cryptography is used
- **Key concept:**
 - ♦ Also TLS-protected inner EAP authentication
 - ♦ But PACs instead X.509 certificates



EAP Fast has been designed by Cisco and can be considered as the successor of LEAP. Other than LEAP, EAP-FAST is a IETF draft. (See draft-cam-winget-eap-fast-01.txt).

Client support has been available since Q4/2004. The main goals of the EAP-FAST design are:

- Strong authentication and session key provision similar like PEAP or EAP-TTLS
- Simple deployment without the use of a PKI
- Fast roaming support in order to allow for VoIP applications (WDS integration)
- Computationally lightweight by using symmetric cryptography

EAP-FAST uses so-called Protected Access Credentials (PACs) instead of certificates. The protocol must facilitate the use of a single strong shared secret by the peer while enabling the servers to minimize the per user and device state it must cache and manage.

PACs



- **First, Protected Access Credentials (PACs) are generated by the authentication server and distributed to the clients**
 - ♦ Either manually ("out-of-band")
 - ♦ Or automatically ("in-band" during "phase 0")
- **PACs consist of a secret and opaque part**
 - ♦ Secret part contains keying material
 - ♦ Opaque part is sent by client to prove that he/she also possesses the secret part

Note: also a "Phase 0" had been specified for in-band provisioning, to provide the peer with a shared secret to be used in secure phase 1 conversation. In phase 0, the Authenticated Diffie-Hellman Protocol (ADHP) can be used for PAC-key exchanges. This phase is independent of other phases; hence, any other scheme (in-band or out-of-band) can be used in the future. The main goal of phase 0 is to eliminate the requirement in the client to establish a master secret every time a client requires network access.

The PAC-Opaque contains the PAC-Key encrypted by a strong key only known to the server and is sent to the server with the TLS ClientHello.

PAC Components (Detailed)



- **1) PAC Key**
 - ♦ 32 byte
 - ♦ Randomly generated by AS
 - ♦ Used as TLS pre-master-secret to establish "phase 1" tunnel
- **2) PAC Opaque**
 - ♦ Variable length field
 - ♦ Sent to AS during phase 1 tunnel establishment
 - ♦ Can only be interpreted by AS
 - ♦ Contains the PAC key and the peer's identity
- **3) PAC Info**
 - ♦ Variable length field
 - ♦ Contains readable information such as authority identity (A-ID), PAC issuer, and PAC-key lifetime

An EAP-FAST authentication server is identified by its Authority Identity (A-ID). This A-ID is unique to each server along with the server master key. If an EAP-FAST session starts, the server sends its A-ID in the EAP-FAST start packet. Based on the A-ID, the EAP-FAST client selects the correct PAC.

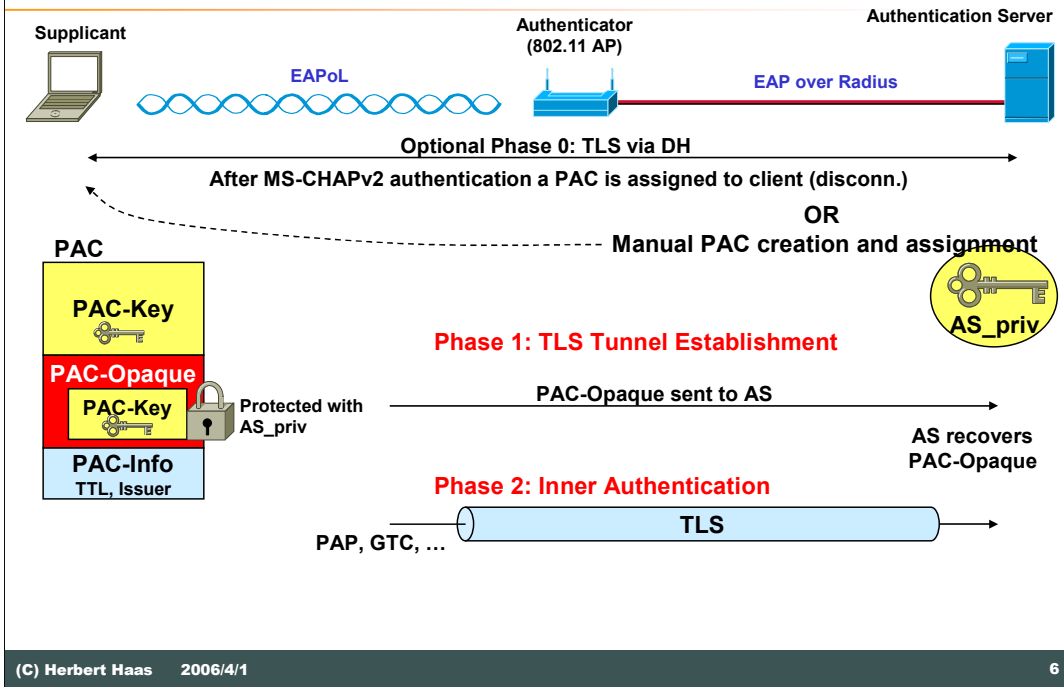
Supports MS-DB, and LDAP-DB. No support for OTP.

Concept



- **Two or three EAP-FAST phases**
 - ◆ Phase 0: (*Optional*) automatic PAC provision
 - ◆ Phase 1: TLS tunnel establishment
 - ◆ Phase 2: Mutual authentication
- **After authentication**
 - ◆ Master Secret Keys (MSKs) are derived
 - ◆ AS can update the client with a fresh PAC key
- **A client may cache multiple PACs to communicate with different authentication servers**

802.1x – EAP-FAST – Details



Note



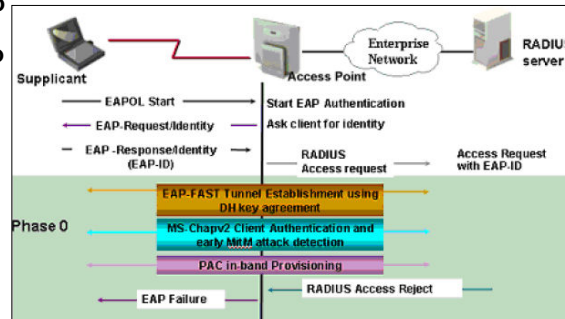
- **No Server States Needed!**
 - ◆ The PAC-opaque is sent by the client and *contains the PAC-key* which is encrypted by ACS's private key
 - ◆ Only *after* receiving the PAC-opaque, the server knows the shared secret and can establish the TLS tunnel with it

One of the main advantages of EAP-FAST is that authentication servers do not have to maintain state information for each client. A client begins authentication by sending the PAC-opaque to the server, which contains the PAC-key encrypted by a strong key only known to the server. That is, upon receiving the PAC-opaque, the server decrypts it and therefore derives the PAC-key

PAC Provisioning

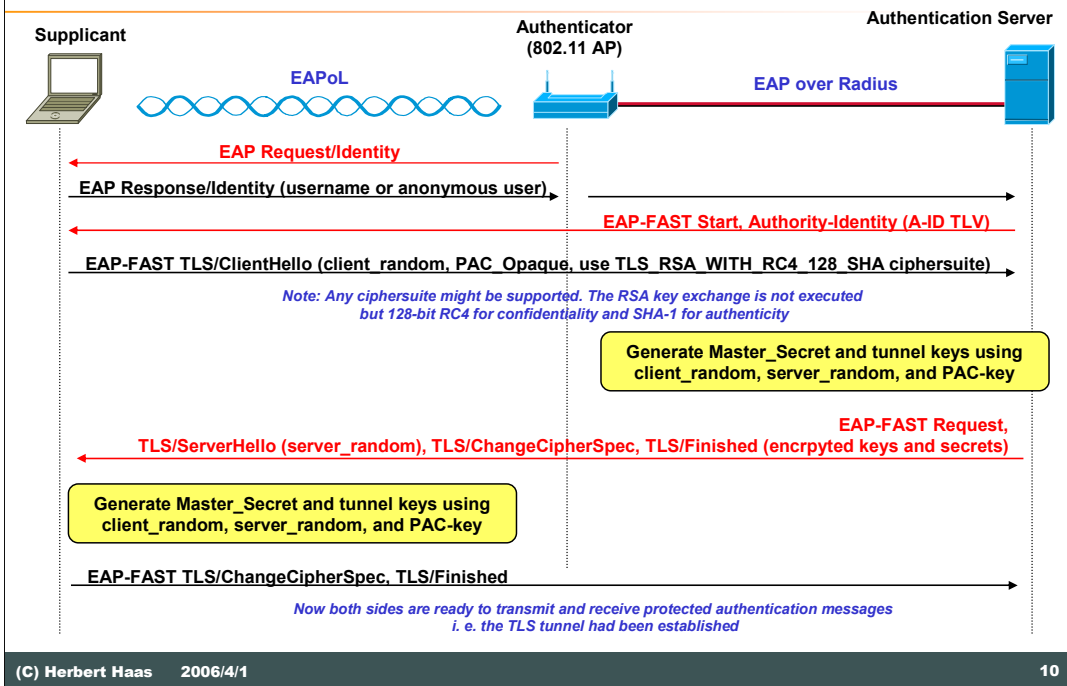


- Phase 0 = PAC auto-provisioning
 - Uses TLS with DH key agreement to establish a secure tunnel
 - Additionally, MS-CHAPv2 is used to authenticate the client and to prevent MITM
 - After the PAC has been successfully provisioned, EAP-FAST is restarted to gain network access
 - Therefore, after a successful PAC provisioning transaction, an EAP *failure* occurs to terminate the EAP-FAST session
 - Afterwards, the newly provisioned PAC can be used to establish an authenticated session
- Manual provisioning ("out-band")
 - May be necessary if a non-Microsoft-format database is used (such as LDAP) which does not support MSCHAPv2 credentials
 - PAC files can be manually generated at the ACS and distributed manually to client devices
 - "Out-of-band" provisioning



Source: Cisco Systems

Phase 1 – Details



(C) Herbert Haas 2006/4/1

10

Note: Since a PAC may be used as a credential for other applications beyond EAP-FAST, the PAC key is further hashed using T-PRF to generate a fresh TLS master_secret. Additionally, the hash of the PAC-key is required to stretch it to the required 48 octet master_secret:

Master_secret = T-PRF(PAC-key, "PAC to master secret label hash", server_random + client_random, 48)

Key material for EAP-FAST tunnel protection:

key_block = PRF(master_secret, "key expansion", server_random + client_random)

("+" denotes concatenation)

In case EAP-FAST authentication employs 128bit RC4 and SHA1, the key_block is partitioned as follows:

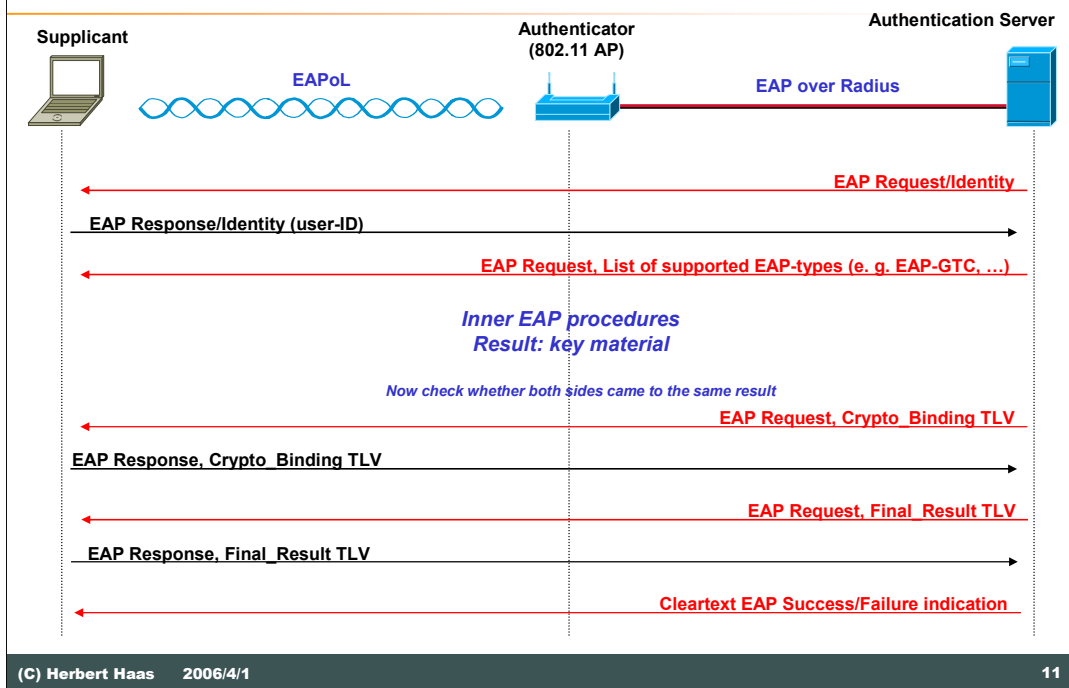
```

client_write_MAC_secret[hash_size=20]
server_write_MAC_secret[hash_size=20]
client_write_key[Key_material_length=16]
server_write_key[key_material_length=16]
client_write_IV[IV_size=0]
server_write_IV[IV_size=0]
session_key_seed[seed_size= 40]
    
```

After phase 2, the MSKs are derived. Part of the MSK is forwarded to the AP by the AS using the RADIUS MS-MPPE attributes (RFC 2548).

Pseudorandom function used as defined in RFC 2246.

Phase 2 – Details



All EAP messages are encapsulated in the EAP Message TLV. Assumption: Phase 1 had been successful, or TLS session had been successfully resumed.

Phase 2 key derivations are used to prove tunnel integrity and to generate session keys. The details depend on the inner EAP method. The inner keying material is always expanded (if necessary) to (at least) 32 octets. The inner keying material (i. e. the result of the inner EAP exchange) is fed into a PRF to generate the MSK.

The phase 2 inner authentication method over EAP-TLV can be EAP-SIM, EAP-OTP, EAP-GTC, or MSCHAPv2.

Additional Facts



- **Client can resume TLS session by sending its session-ID (in a ClientHello)**
 - ♦ Bypass inner EAP conversation
 - ♦ But server must cache client's session-ID, master_secret, and CipherSpec
- **EAP-FAST supports single sign-on (SSO) using username and password during Windows networking logon**
- **Seamless migration from LEAP to EAP-FAST possible**
 - ♦ Similar AP settings
 - ♦ ACU reconfiguration via ACAT
- **WPA is also supported**