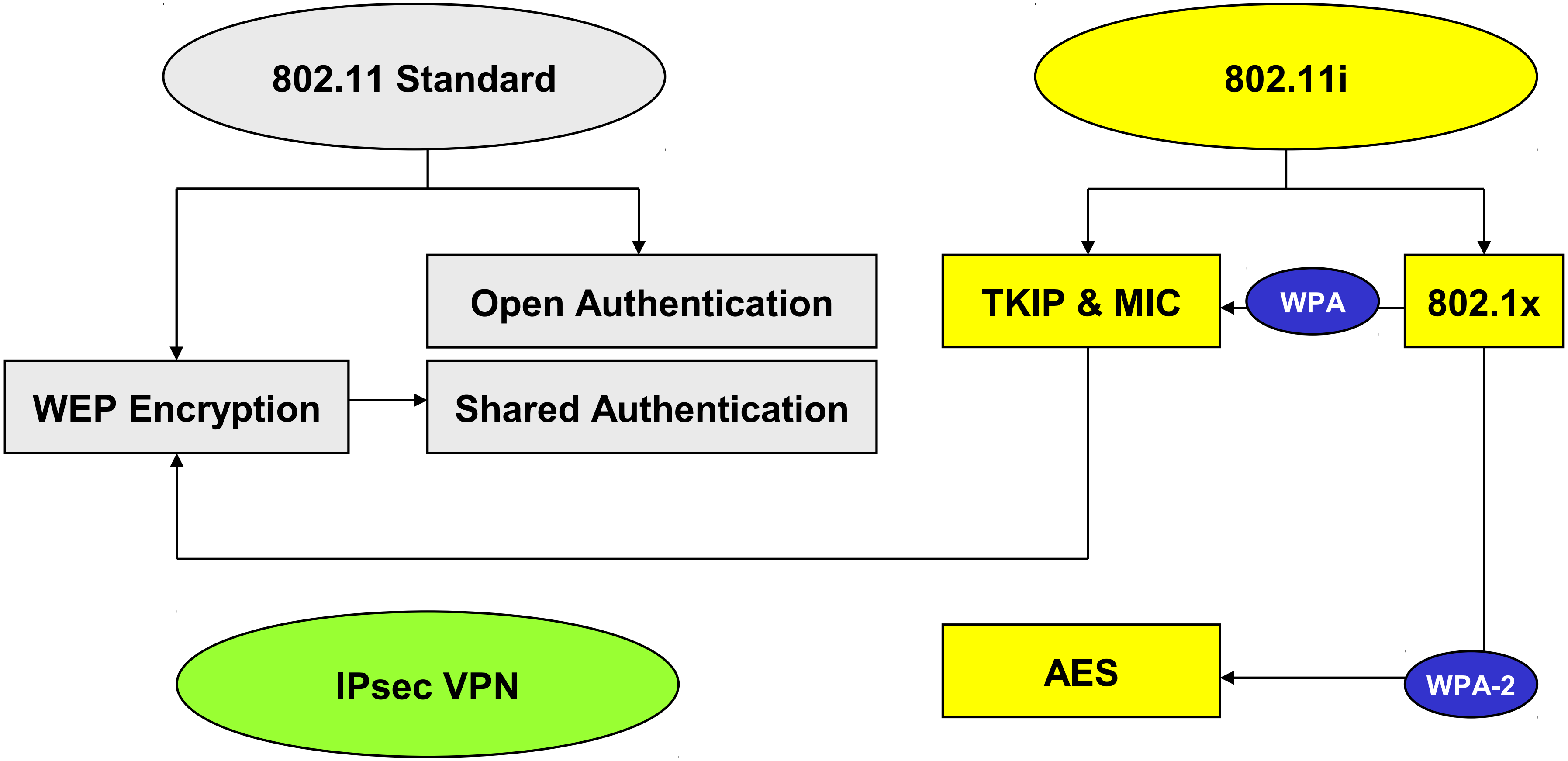
# WLAN

Security Summary

# Threat Summary

- Simple eavesdropping
  - Radio broadcast
  - Reduce TX powers!
  - Encryption (WEP, TKIP, AES, IPsec)
- Authentication
  - Shared secrets vs. stolen devices, large nets
  - Centralized AAA => 802.1x
  - Mutual authentication (Rogue APs)
- DoS Attacks
  - Physical jamming
  - Difficult to prevent (shielding, directional antennas)

# WLAN Security Overview



# WEP Problems

# Intro

- Wireless LAN is a perfect media for attackers
  - Sniffers easily remain undetected
  - Outdoor attacks
  - Simple DoS attacks through jamming
- Vulnerabilities found in initial standards
  - Authentication / Encryption / Integrity
  - Centralized management of user credentials
- “Mobile devices” => frequent hardware theft
- Rogue APs often remain undetected
  - Mutual auth required
- Interoperability of security features of different vendors still in question (nevertheless WPA)
- Lots of cracker tools available (WEPCrack, AsLeap, …)
- 2002/2003: 66% of WLANs unprotected (but better security awareness in 2004)

# RC4 Facts

- Simple and fast stream cipher
  - Variable key lengths (1-256 bytes)
  - 15 times faster than 3DES
    - 8-16 operations per output byte
  - Also used by SSL/TLS
- Designed 1987 by Ron Rivest for RSA Security
  - Kept as trade secret by RSA Security but leaked out in 1994
- Period is larger than $10^{100}$ !!!

# How RC4 Works

```
for i = 0 to 255 do
  S[i] = i;
  T[i] = K[i mod keylen];
```

Initialize S[0]..S[255] with ascending numbers.
Initialize T[0]..T[255] with the key K (If keylen < 256 then repeat K as often as necessary).

```
j = 0;
for i = 0 to 256 do
  j = (j + S[i] + T[i]) mod 256;
  Swap (S[i], S[j]);
```

Use T to produce initial permutation of S.
Hereby go from S[0] to S[255] and swap each S[i] with another byte dictated by T[i].

After that, S still contains all numbers from 0 to 255 but in a permutated order.

```
i, j = 0;
while (1)
  i = (i + 1) mod 256;
  j = (j + S[i]) mod 256;
  Swap (S[i], S[j]);
  t = (S[i] + S[j]) mod 256;
  k = S[t];
```

Now again swap S[i] with another byte in S, but this time it is dictated by S itself (the key is no longer used).

After S[255] is reached, repeat again with S[0], as long as there are bytes to encrypt or decrypt.

XOR byte k with plaintext byte or ciphertext byte for encryption or decryption respectively.

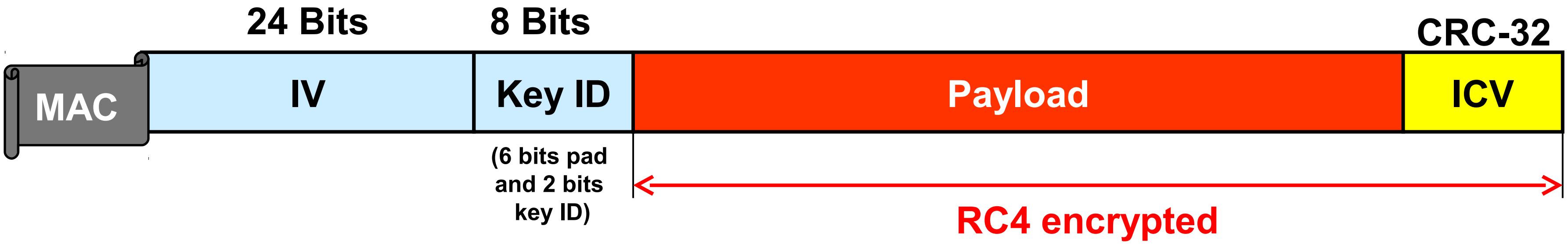
# General Stream Cipher Issues

- Every stream cipher is supposed to produce a good pseudorandom "keystream"
  - This is the idea of a "one-time pad"
- The keystream is XORed with the plaintext
- This method is secure *if*
  - The keystream-generator has high entropy (i. e. really random)
  - Each keystream is only used once

# Wired Equivalent Privacy (WEP)

- Only encryption method of the 802.11 standard
  - Used for privacy, integrity and authentication
- Shared key method
  - Either one static key
  - Or short list of dynamic keys (up to four)
- Key lengths:
  - 40 bit (default, aka "64 bit" with IV)
  - Optionally 104 (or "128" bit with IV)
- No key distribution method defined(!)

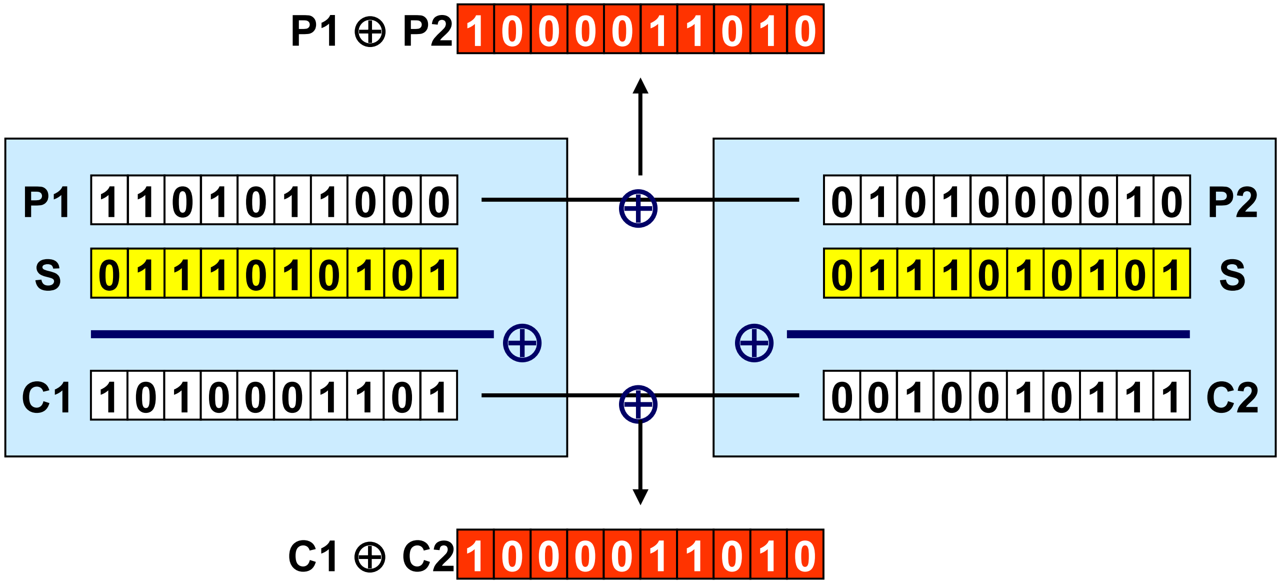
# Basic Principle



- Payload is XORed with a RC4-generated pseudorandom keystream K
  - S depends on shared key and 24 bit Initialization Vector (IV)
  - Ciphertext C = Plaintext P ⊕ Keystream K

# WEP – Design Flaw in Detail

- **The Problem:**
  - **XOR operation eliminates two identical terms!**
  - **If same S is used on different plaintexts, then**
    - **C1=S ⊕ P1 and C2=S ⊕ P2**
    - **C1 ⊕ C2 = P1 ⊕ P2**
    - **Same keystream S cancels out!**
  - **If P1 is known then P2 can be easily calculated!**



P1 ⊕ P2 1 0 0 0 0 1 1 0 1 0

P1 1 1 0 1 0 1 1 0 0 0

S 0 1 1 1 0 1 0 1 0 1

⊕

C1 1 0 1 0 0 0 1 1 0 1

0 1 0 1 0 0 0 0 1 0 P2

0 1 1 1 0 1 0 1 0 1 S

⊕

0 0 1 0 0 1 0 1 1 1 C2

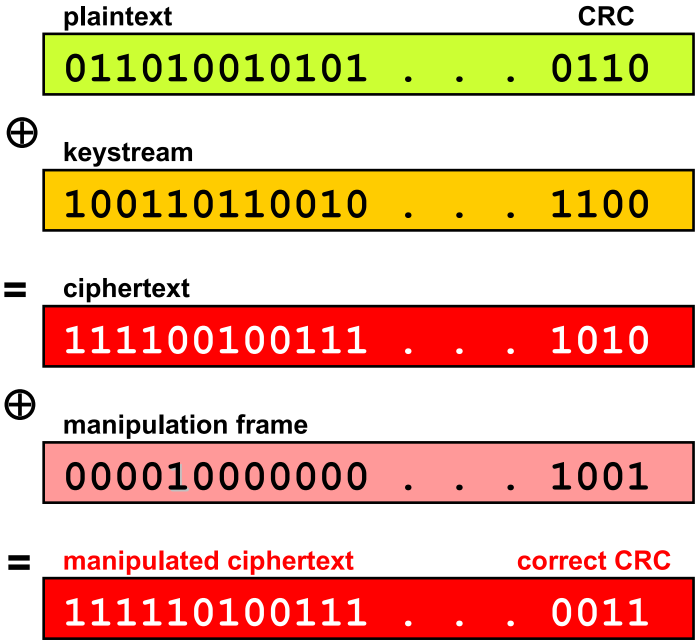
C1 ⊕ C2 1 0 0 0 0 1 1 0 1 0

# IV Collisions

- Keystream should change for each packet
  - Assures that same plaintexts result in different Ciphertext
  - 802.11 does not specify how to pick IVs
  - Many implementations reset IV to zero at startup and then count up
- Only $2^{24}$ IV choices → Collisions will occur !!!
  - Attacker could maintain a "codebook" of all possible S
  - 1500 byte × $2^{24}$ = 24 GByte
  - Matter of hours only
- Shared key length does not hamper the attack!

# Integrity Vulnerability

- Encrypted CRC is used to check integrity
- But CRC is linear:
  - $CRC(X \oplus Y) = CRC(X) \oplus CRC(Y)$
- Thus payload bits can be manipulated, because
  - $RC4^{K}(X \oplus Y) = RC4^{K}(X) \oplus Y$
  - $RC4^{K}(CRC(X \oplus Y)) = RC4^{K}(CRC(X)) \oplus CRC(Y)$
- Attacker can easily modify known bytes of packets (at least L3/L4 header structures are known)



plaintext CRC

011010010101 . . . 0110

⊕ keystream

100110110010 . . . 1100

= ciphertext

111100100111 . . . 1010

⊕ manipulation frame

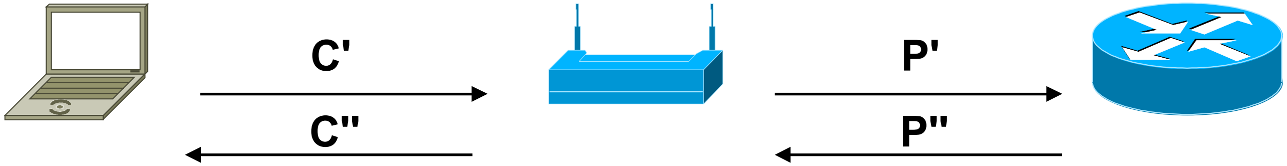
000010000000 . . . 1001

= manipulated ciphertext correct CRC

111110100111 . . . 0011

# Bit-Flipping Attack Example

- Attacker catches and manipulates encrypted frame, updates ICV
- AP decrypts frame, validates ICV and forwards frame
- Router detects fault and sends predictable error message
- Keystream = C'' + P''



# Arbaugh Attack

- Allows to arbitrarily expand a known keystream of size n
  - Easily done with known messages (e. g. DHCP discoveries)
- Create messages of size n-3 and encrypt it with the known keystream
- Only the last byte (4th CRC byte) is not encrypted: trial and error!
- On average only 128 trials necessary for every additional byte!

# Attacks Summary (1)

- ***Keystream reuse (IV collisions)***
  - **Dictionary-building attacks**
  - **Allows real-time automated decryption of all traffic**
- ***Bit-flipping attacks***
  - **Attacker intercepts WEP-encrypted packet, flips bits recalculates CRC and retransmits forged packet to AP with same IV**
  - **Because CRC32 is correct, AP accepts and forwards frame**
  - **Layer 3 end device rejects and sends a predictable response**
  - **AP encrypts response and sends it to attacker**
  - **Attacker uses response to derive key**

# Attacks Summary (2)

- ***Fluhrer, Mantin, Shamir (FMS) attack on RC4***
  - RC4 key scheduling is insufficient
    - The beginning of the pseudorandom stream should be skipped, otherwise some IV values reveal information about the key state
  - Key can be recovered after several million packets
  - 'WEPplus' = WEP with avoidance of weak IVs
- ***KoreK Attack***
  - Packet manipulation, reinjection and CRC analysis
  - Key can be recovered after several 100,000 packets
- ***Arbaugh Attack***
  - Calculate arbitrary additional bytes on a known but short keystream

# Interim Solutions: TKIP and MIC

# 802.11i

- Two new network types
  - Transition Security Network (TSN)
  - Robust Security Network (RSN)
- An RSN only allows devices using TKIP/Michael and CCMP
- A TSN supports both RSN and pre-RSN (WEP) devices
  - Problem: broadcast packets have to be transmitted with the weakest common denominator security method
  - Consider a single client only supporting WEP

# 802.11i

Pre-standard 802.11i
(WPA)

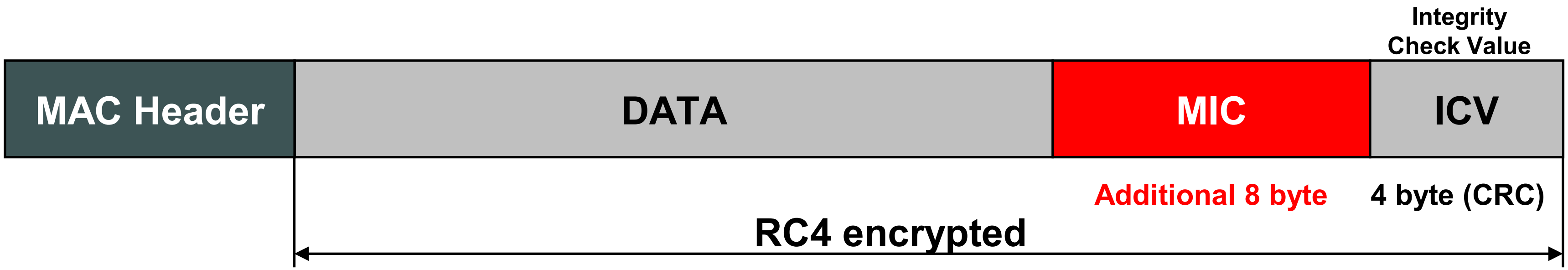
- Message Integrity Check (MIC)
  - Nonlinear algorithm
- Temporal Key Integrity Protocol (TKIP or "WEP2")
  - Also uses RC4-based WEP without the known flaws
    - Per-packet keys through IV mixing
    - Replay protection
  - Essentially a patch for WEP

Ratified 802.11i
(WPA2)

First WPA2 certifications
already since 1st Sept 2004

- Counter Mode CBC MAC (CCMP)
  - = AES + CBC-MAC
  - Replaces WEP !!!
    (requires new HW support)

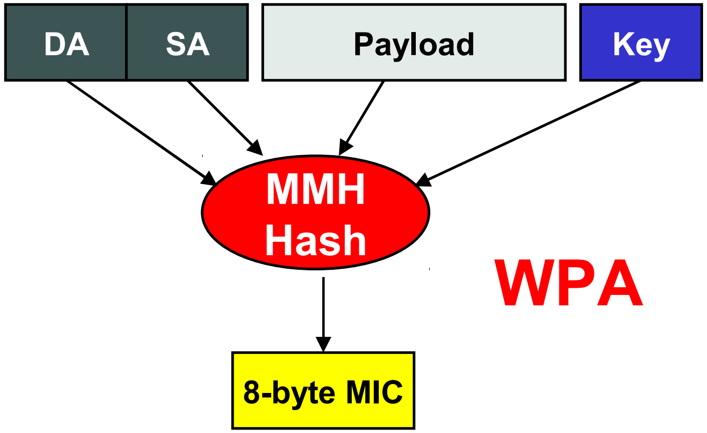
# MIC (as used by WPA)



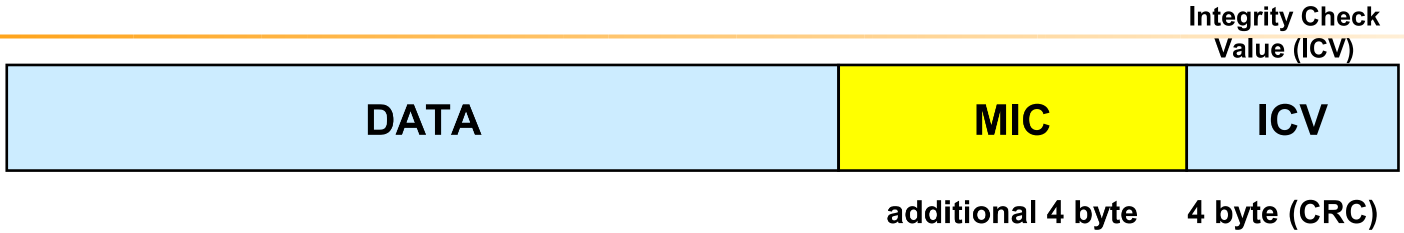
- Encrypted checksum
  - => Nonlinear function now
- Uses "Michael" algorithm
  - Much more lightweight than MD5 or SHA
- Uses separate 64-bit key
  - Data Integrity Key (DIK) derived from PTK after WPA key management
  - AP and STA use different MIC keys (128-bit DIK is split)

# MIC Problems

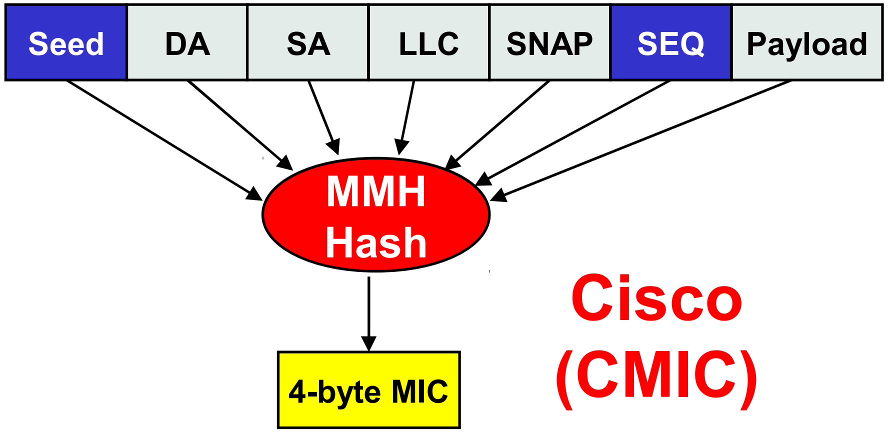
- Michael algorithm
  - Provides security level of only 20 bit strength
  - Attacker can construct forgery after approx 2^19 tries (520,000 frames)
- MIC Countermeasures
  - Upon two MIC failures within 60 seconds, this AP disassociates *all* stations for at least 60 seconds and erases current keys in use
  - So attacker forgery trials become nearly impossible
  - Typically turned OFF (DoS!!!)



# Cisco MIC (CMIC)

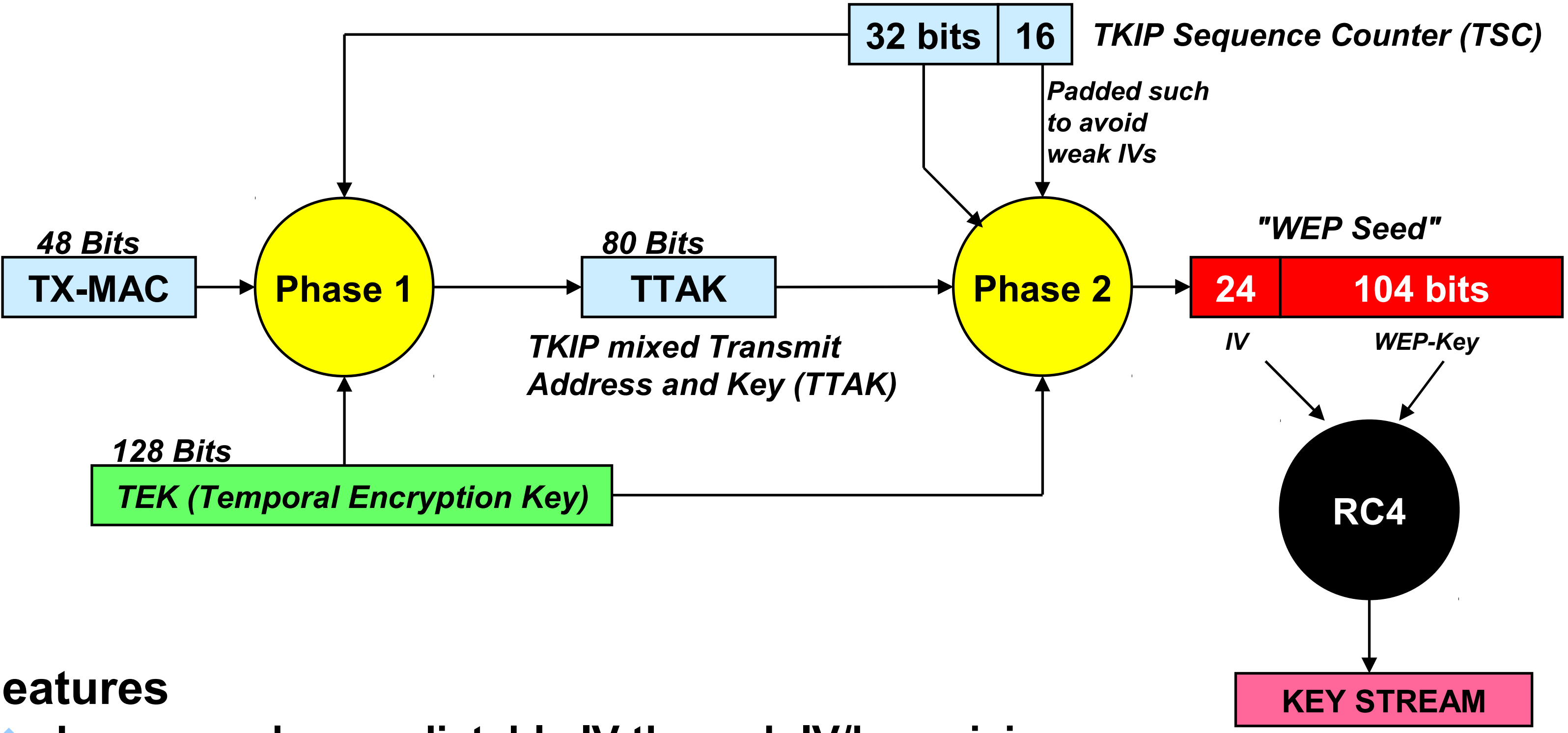


- Uses a seed value as pseudo-key
- Uses sequence number (AP verifies order)



Cisco (CMIC)

# TKIP (As used by WPA)

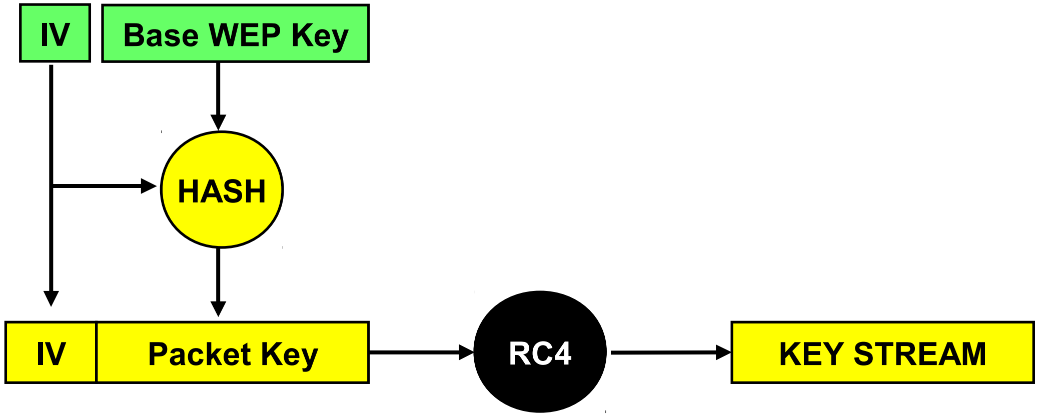

32 bits
16
TKIP Sequence Counter (TSC)
Padded such to avoid weak IVs
48 Bits
TX-MAC
Phase 1
80 Bits
TTAK
Phase 2
"WEP Seed"
24
104 bits
IV
WEP-Key
TKIP mixed Transmit Address and Key (TTAK)
128 Bits
TEK (Temporal Encryption Key)
RC4
KEY STREAM

- Features
  - Longer and unpredictable IV through IV/key mixing
  - Encrypted replay protection number (TSC)
- WPA TKIP
  - 48 bit IV, includes MAC
  - Fast S-box mixer
  - Fresh session keys on every association

# TKIP Details

- Phase 1
  - The high-order 32 bits of the TSC are combined with the TA and the first 80 bits of the TEK.
  - This phase of the key mixing is an iteration involving inexpensive addition, XOR, and AND operations, plus an S-box lookup reminiscent of the RC4 algorithm. These were chosen for their ease of computation on low-end devices such as APs.
  - Phase 1 produces an 80-bit value called TKIP mixed Transmit Address and Key (TTAK). Note that the only input of this phase that changes between packets is the TSC. Because it uses the high-order bits, it only changes every 64K packets.
  - Phase 1 can thus be run infrequently and use a stored TTAK to speed up processing. The inclusion of the transmitter's MAC address is important to allow a pair of stations to use the same TEK and TSC values and not repeat RC4 keys.
- Phase 2
  - Now the TTAK from phase 1 is combined with the full TEK and the full TSC.
  - This phase again uses inexpensive operations, including addition, XOR, AND, OR, bit-shifting, and an S-box.
  - The output is a 128-bit WEP seed that will be used as the RC4 key in the same manner as traditional WEP.
  - In the phase 2 algorithm, the first 24 bits of the WEP seed are constructed from the TSC in a way that avoids certain classes of weak RC4 keys.

# Cisco TKIP ("CKIP")



- Simple proprietary solution
- Still uses 24 bit IV but calculates per-packet WEP keys from IV
  - Hash-based mixer

# Security

- Against rumors, TKIP is reasonably safe!
  - For each packet, the 48-bit IV is mixed with the 128-bit PTK to create a 104-bit RC4 key
    - There is practically no statistical correlation
    - Estimated one weak-IV per century (!)
  - Countermeasures against traffic re-injection
    - Sequence numbers + MIC
  - Robust 4-way handshake
- Only problem: WPA-PSK
  - Which uses a specified passphrase to PMK mapping => good passphrase required !!!
  - Otherwise dictionary attack possible

# AES and CCMP

# 802.11i

Pre-standard 802.11i – TSN (WPA)

- Message Integrity Check (MIC)
  - Nonlinear algorithm
- Temporal Key Integrity Protocol (TKIP or "WEP2")
  - Also uses RC4-based WEP without the known flaws
    - Per-packet keys through IV mixing
    - Replay protection
  - Essentially a patch for WEP

Ratified 802.11i – RSN (WPA2)

First WPA2 certifications already since 1st Sept 2004

- Counter Mode CBC MAC (CCMP)
  - = AES + CBC-MAC
  - Replaces WEP !!! (requires new HW support)

# WPA2 aka 802.11i

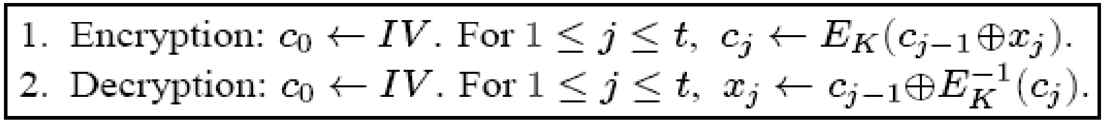
- Exactly the same as WPA1 except...
  - CCMP (AES in counter mode) instead of RC4
  - HMAC-SHA1 instead of HMAC-MD5 for the EAPoL MIC
- Against rumors WPA2 is only a LITTLE better than WPA1
  - But neither will be cracked in the near future !!!

# 802.11i: CCMP – Overview

- AES for data encryption (privacy)
  - 128-bit block cipher
  - No per-packet keying needed
  - HW-realization recommended
  - Key-life determined by 48-bit IV
- AES requires a feedback mode
  - To avoid the risks associated with the trivial Electronic Codebook (ECB) mode
    - Repeating patterns are not hidden
    - Not recommended for messages longer than one block !
- The IEEE is still deciding which feedback mode to standardize for AES encryption – two choices:
  - Counter Mode CBC MAC (CCM)
    - Provides encryption, authenticity and integrity
    - Applied on both header and data
    - IV also used to prevent replay attacks
    - WLAN's current favourite
  - Offline Code Book (OCB) mode
    - Problem: patented
    - Also supported by some WLAN vendors

# Cipher Block Chaining (CBC)

- No patent
- Encryption and MAC use different nonces
  - Collision attacks possible but sufficient mitigation when key management provides frequent key changes
- Identical ciphertext blocks result only when:
  - Same key and
  - Same plaintext and
  - Same IV is used
- CBC is self-synchronizing
  - If an error (including loss of one or more entire blocks) occurs in block $c_j$ but not $c_{j+1}$, then $c_{j+2}$ is correctly decrypted to $x_{j+2}$.



1. Encryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t,\ c_j \leftarrow E_K(c_{j-1} \oplus x_j)$.
2. Decryption: $c_0 \leftarrow IV$. For $1 \leq j \leq t,\ x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j)$.

# Counter Mode (CCM)

- Instead of directly encrypting the data only a counter is encrypted
- Message is then XORed with this encrypted counter
- Counter = nonce (SQNR, Source-MAC, Priority fields)

# Offset Code Book (OCB)

- Patented
- Combines authentication and encryption
  - Slightly faster than CBC encryption
  - More prone to collision attacks than CBC-MAC
- If a particular collision on 128-bit values occurs, then an attacker can modify the message without being detected by the OCB authentication function
  - Weak authentication algorithm – uses same nonce for encryption and authentication
  - In order to limit the probability of a successful forgery attempt to less than 2^-64 change the key after 2^32 blocks of data
  - Indeed strong enough for many people but does not justify 128-bit AES as successor of DES

# OCB Algorithm

**Convention: Message M, Key K, Nonce N**

**Define** $L := E_K(0)$, $R := E_K(N \oplus L)$ **from which the offset** $Z_i := \gamma_i \cdot L \oplus R$ **follows.**

**Then the message is split into $M_1$, …, $M_m$, where only $M_m$ is typically a non-128 bit block. The messages $M_1$, … $M_{m-1}$ are encrypted as follows:**

$$X_i := M_i \oplus Z_i$$
$$Y_i := E_K(X_i)$$
$$C_i := Y_i \oplus Z_i$$

**While $M_m$ is encrypted using μ denoting the length of this block:**

$$X_m := \mu \oplus x^{-1} \cdot L \oplus Z_m$$
$$Y_m := E_K(X_m)$$
$$C_m := M_m \oplus \text{first-}\mu\text{-bits}(Y_m)$$

**The authentication is performed in two steps:**

$$S := M_1 \oplus \cdots \oplus M_{m-1} \oplus C_m0^* \oplus Y_m$$

**… "Checksum"**

$$T := \text{first-}\tau\text{-bits}(E_K(S) \oplus Z_m)$$

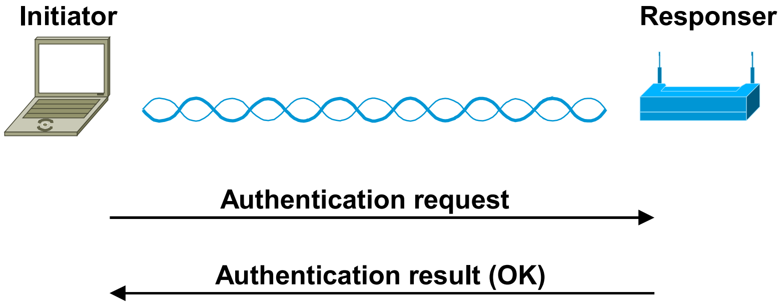
**… "MAC Tag" of arbitrary length, depending on security vs. transmission cost trade-off. Typically 32..80 (documentation)**

**$C_m0^*$ … last ciphertext block padded with zeros to full 128 bit length**

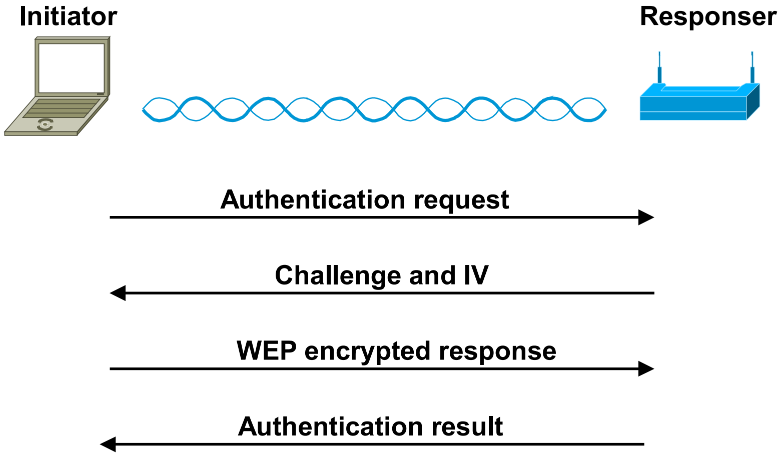
# 802.11 Standard Authentication

# 802.11 Standard Authentication Methods

- **Open System Authentication**
  - Anyone is granted access
  - Ideal for transient users
  - Default method
  - All frames sent in clear, even when WEP is enabled

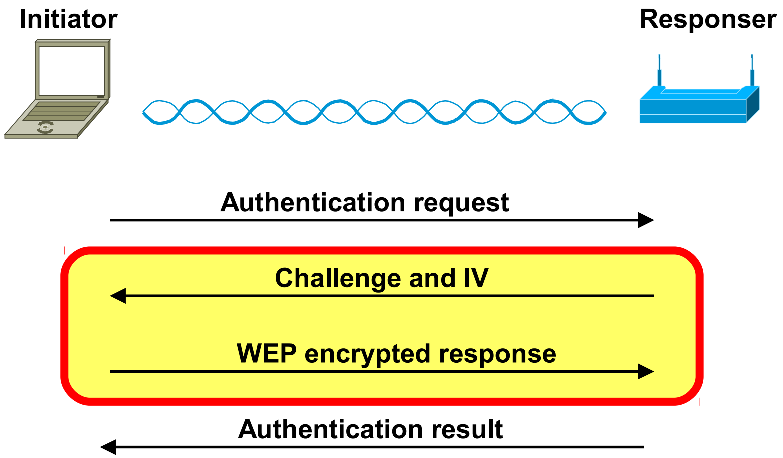


- **Shared Key Authentication**
  - Relies on WEP algorithm
  - Every user has same shared key—and same as AP
  - Only client device authentication
  - User is not authenticated (device theft critical)
  - AP is not authenticated (!)
  - Vulnerable…



# Shared Key Authentication

- Attacker captures 2nd and 3rd authentication message and has
  - Plaintext P (the challenge)
  - Ciphertext C = RC4K (P)
- The keystream is simply
  S = C ⊕ P
- Other fields than the challenge are known a priori
  - Have always the same value in each authentication process
- Possessing S, an attacker can correctly respond to each challenge
- Never use Shared Key Authentication !!!



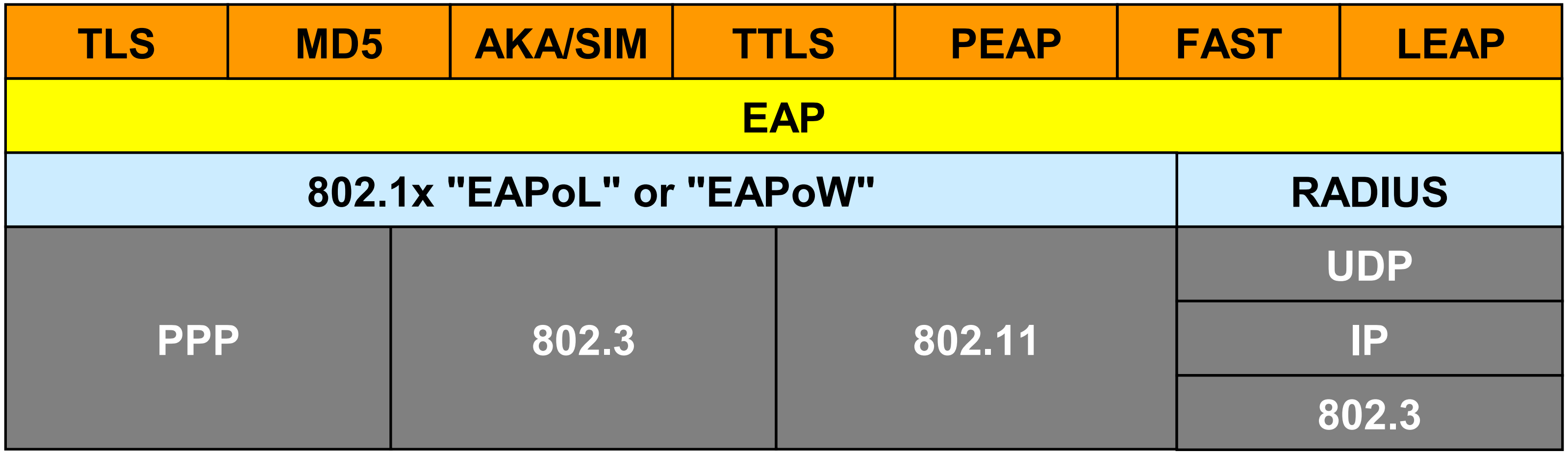
# 802.1x and EAP Authentication

# 802.1x Authentication – Intro

- Port-based network access control method utilizing IETF's Extensible Authentication Protocol (EAP)
  - Supports mutual authentication between client and AP
- Dynamic WEP/TKIP key distribution and refresh
  - Only for unicast traffic
    - Each client has its own key—as long as AP has enough key slots
    - Session lifetime
  - But static and shared broadcast key
    - Either pre-configured or automatically assigned after authentication
- Centralized user credential management via RADIUS
  - Various client credentials supported
- (Fast) L2 roaming support (*possible*)

# What is EAP?

- Extensible: allows to develop and deploy new authentication protocols easily
  - No SW update on authenticator (AP) needed
  - Only supplicant and AS server need to be updated
- See RFC 2284



TLS | MD5 | AKA/SIM | TTLS | PEAP | FAST | LEAP

EAP

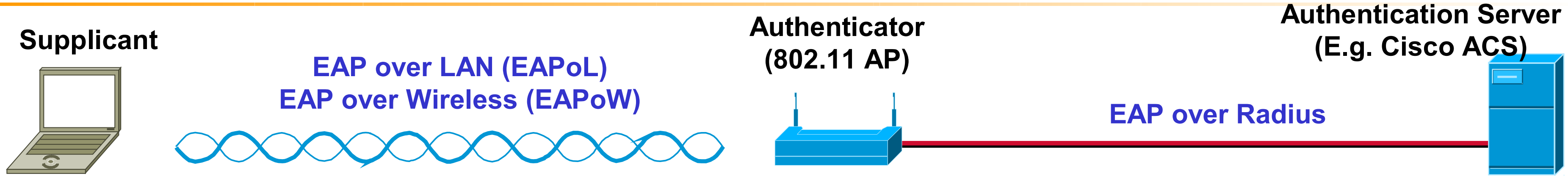
802.1x "EAPoL" or "EAPoW" | RADIUS

UDP

IP

PPP | 802.3 | 802.11 | 802.3

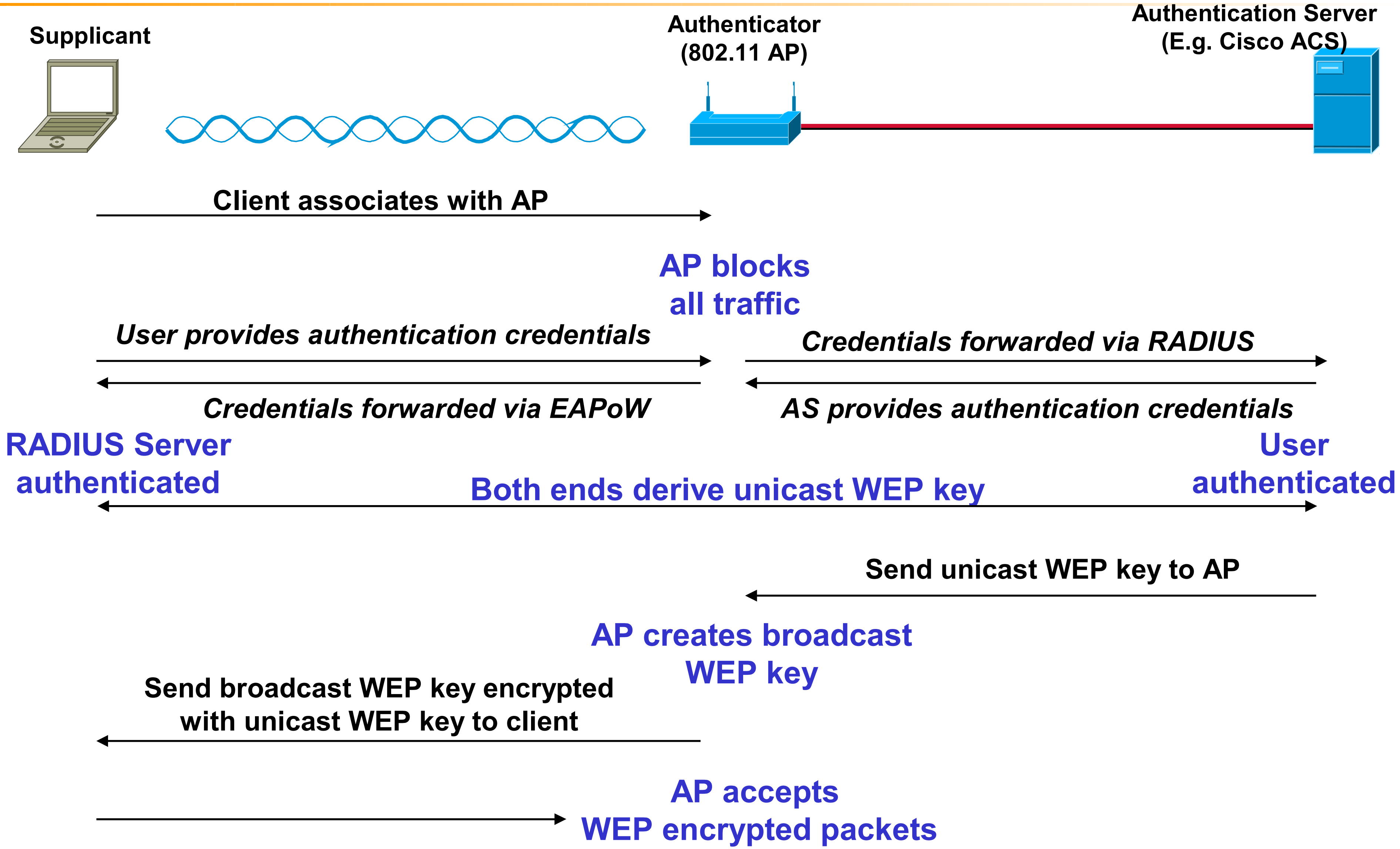
# 802.1x – Protocol Layers



| | | | |
|---|---|---|---|
| EAP's Authentication Method | | | |
| EAP | | | |
| 802.1x | 802.1x | RADIUS | RADIUS |
| | | UDP/IP | UDP/IP |
| 802.11 | 802.11 | 802.3 | 802.3 |

- Authenticator (AP) blocks access until client is authenticated
  - Only accepts Ethertype 0x888E (EAPoL)
- 802.1x frames are sent to multicast DA = 01-80-C2-00-00-03
- Authenticator translates 802.1x to UDP/IP

# 802.1x – EAP Concept



Supplicant

Authenticator
(802.11 AP)

Authentication Server
(E.g. Cisco ACS)

Client associates with AP

AP blocks
all traffic

*User provides authentication credentials*

*Credentials forwarded via RADIUS*

*Credentials forwarded via EAPoW*

*AS provides authentication credentials*

RADIUS Server
authenticated

User
authenticated

Both ends derive unicast WEP key

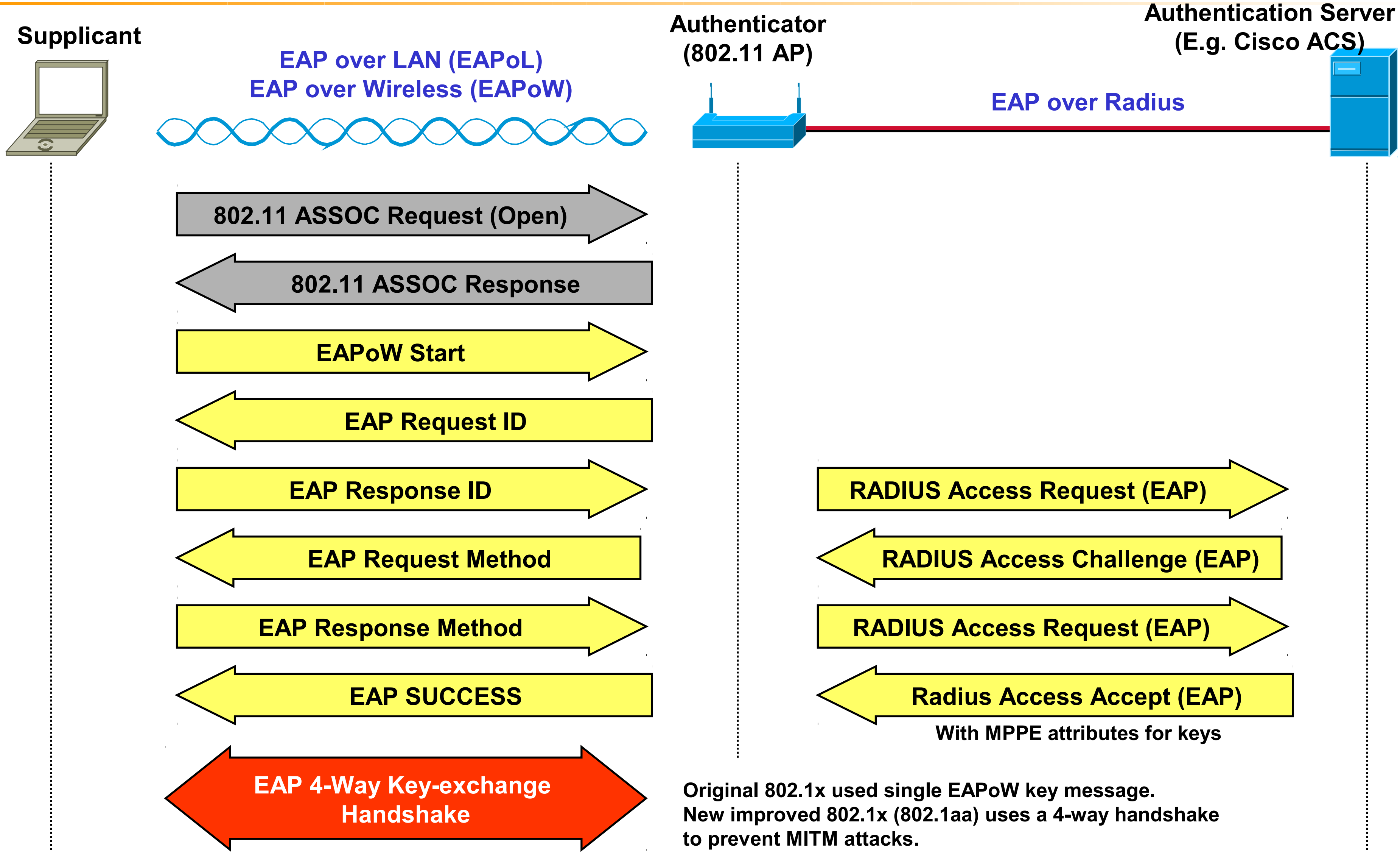
Send unicast WEP key to AP

AP creates broadcast
WEP key

Send broadcast WEP key encrypted
with unicast WEP key to client

AP accepts
WEP encrypted packets

# 802.1x – EAP Protocol



Supplicant

EAP over LAN (EAPoL)
EAP over Wireless (EAPoW)

Authenticator
(802.11 AP)

Authentication Server
(E.g. Cisco ACS)

EAP over Radius

802.11 ASSOC Request (Open)

802.11 ASSOC Response

EAPoW Start

EAP Request ID

EAP Response ID

RADIUS Access Request (EAP)

EAP Request Method

RADIUS Access Challenge (EAP)

EAP Response Method

RADIUS Access Request (EAP)

EAP SUCCESS

Radius Access Accept (EAP)

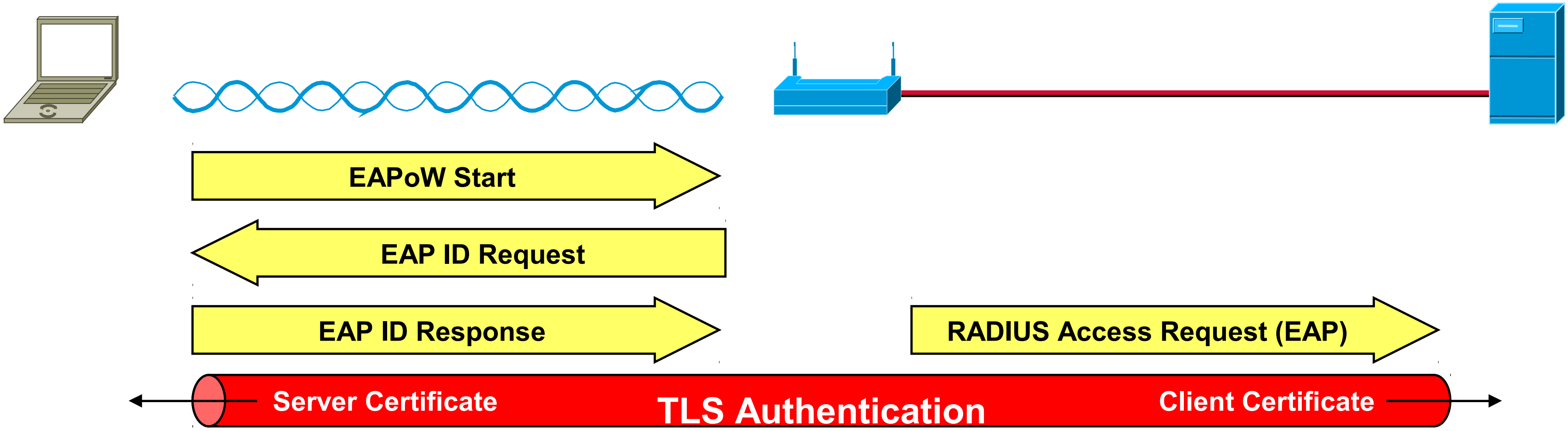
With MPPE attributes for keys

EAP 4-Way Key-exchange Handshake

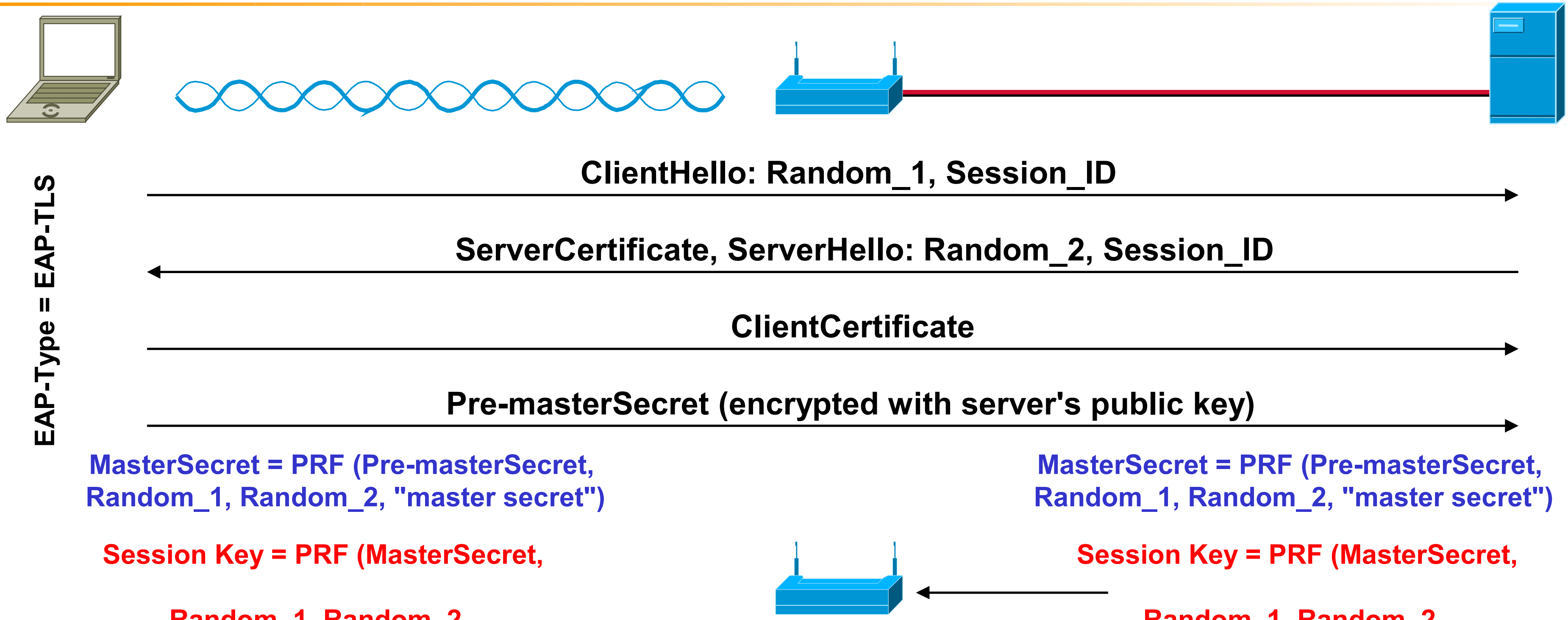
Original 802.1x used single EAPoW key message.
New improved 802.1x (802.1aa) uses a 4-way handshake
to prevent MITM attacks.

# 802.1x – EAP-TLS (1)

- First secure 802.1x realization, EAP method 13 (RFC 2716)
- Relies on Transport Layer Security (TLS)
  - Successor of SSL version 3.0, adopted by IETF
  - Both clients and AS authenticated via certificates
  - *Only TLS authentication and tunnel establishment procedure (tunnel not used)*
  - TLS also used to derive link-layer key between endpoints
- Problems:
  - Client identity is not protected
  - No fast session reconnection
  - Need for PKI (practical: certificate stored in token card or similar)
- Prerequisite for WPA certification
  - Until May 2005 the only required EAP method for WPA



# 802.1x – EAP-TLS (2)



EAP-Type = EAP-TLS

ClientHello: Random_1, Session_ID

ServerCertificate, ServerHello: Random_2, Session_ID

ClientCertificate

Pre-masterSecret (encrypted with server's public key)

MasterSecret = PRF (Pre-masterSecret, Random_1, Random_2, "master secret")

MasterSecret = PRF (Pre-masterSecret, Random_1, Random_2, "master secret")

Session Key = PRF (MasterSecret, Random_1, Random_2, "client EAP encryption")

Session Key = PRF (MasterSecret, Random_1, Random_2, "client EAP encryption")

Authenticator MAY choose subsequent keying material (encryption keys, MAC-keys, and IV) from this session key (for example using the 1st 32-byte block as encryption key, the 2nd 32-byte block as MAC-key and so on…)

- After each re-authentication a new session key can be generated based on the same master key
- Note: TLS details omitted in the picture
  - Such as record details (server_key_exchange, change_cipher_spec, …)

# 802.1x – LEAP

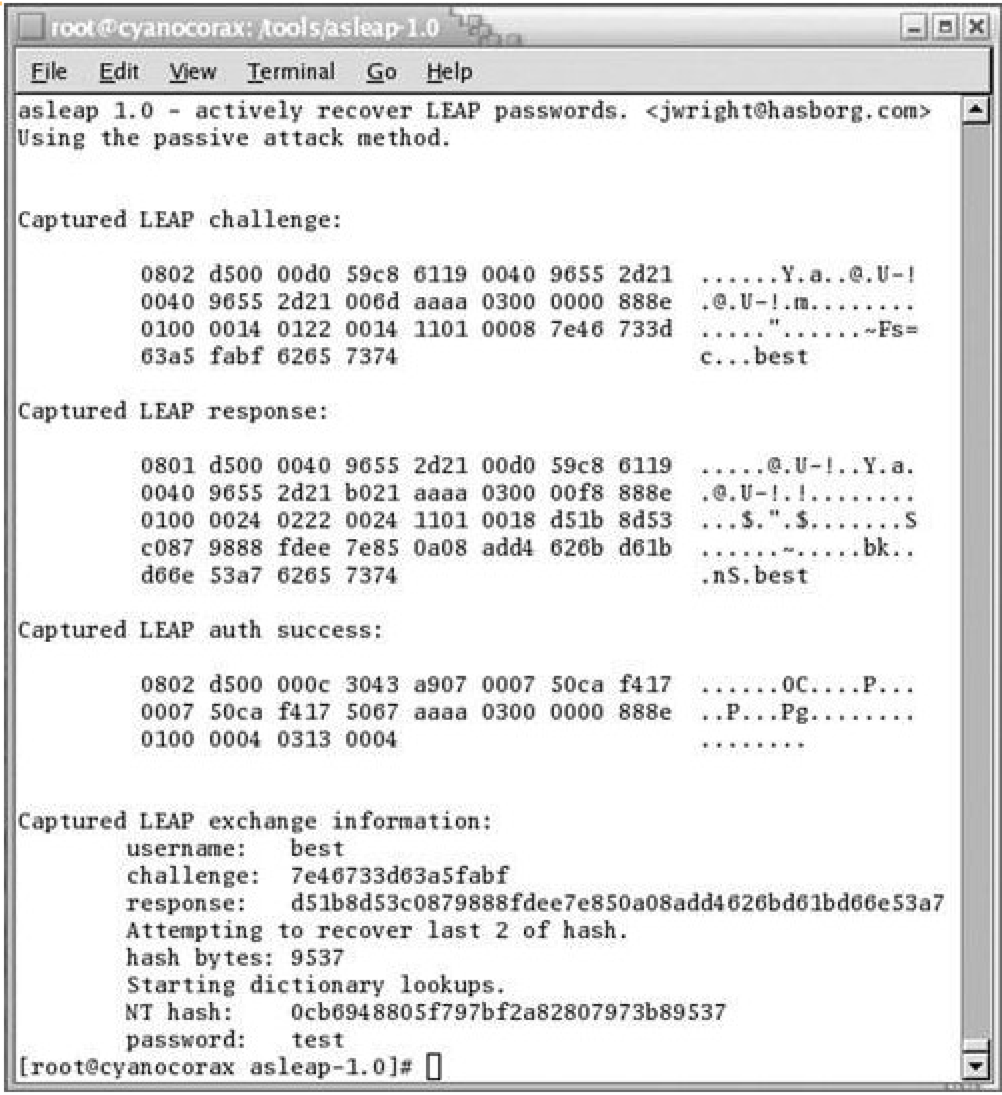
- Cisco's lightweight implementation
- Fast Secure Roaming (< 150 ms)
- Challenge-response based on shared secrets
    - Implemented similar as MS-CHAPv2 (two stage MD4 hashing of passwords)
- Can utilize existing Windows NT Domain Services authentication databases as well as Windows 2000 Active Directory databases
    - No support for LDAP and NIS
- Drivers for Windows 95, 98, Me, 2000, NT and XP and uses the Windows logon as the Cisco LEAP logon
- Also Linux and Mac support
- Vulnerable to dictionary attacks
    - Secure if strong passwords are *enforced* (10 chars at minimum)

# LEAP / MSCHAPv2 Flaws

- AS sends 8 byte challenge
- Client encrypts challenge 3 times using NT hash of the password as DES seed (=key)
  - DES requires a 7 byte seed value in this algorithm
  - So client splits 16 byte NT hash into three portions:
    - Seed1 = B1 .. B7
    - Seed2 = B8 .. B14
    - Seed3 = B15, B16, 0x00, 0x00, 0x00, 0x00, 0x00
- Flaw: third DES output is cryptographically weak, leaving only 2^16 possible permutations
- After B15 and B16 are known, we can significantly reduce the number of potential matches in our dictionary file, using the known 2 bytes of the user's hash as a keying mechanism

# Asleap

- Offline attack on LEAP
- Principle:
  - LEAP performs unencrypted MSCHAPv2 (challenge-handshake)
  - Asleap captures challenge and encrypted reply and performs an offline dictionary attack
- Written by Joshua Wright
- http://asleap.sourceforge.net/
- Also see Leapcrack


root@cyanocorax: /tools/asleap-1.0
File Edit View Terminal Go Help

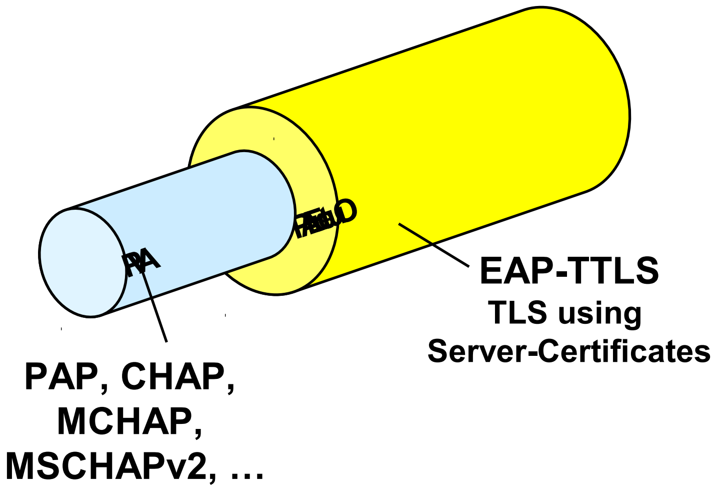
```
asleap 1.0 - actively recover LEAP passwords. <jwright@hasborg.com>
Using the passive attack method.


Captured LEAP challenge:

         0802 d500 00d0 59c8 6119 0040 9655 2d21  ......Y.a..@.U-!
         0040 9655 2d21 006d aaaa 0300 0000 888e  .@.U-!.m........
         0100 0014 0122 0014 1101 0008 7e46 733d  ....."......~Fs=
         63a5 fabf 6265 7374                      c...best

Captured LEAP response:

         0801 d500 0040 9655 2d21 00d0 59c8 6119  .....@.U-!..Y.a.
         0040 9655 2d21 b021 aaaa 0300 00f8 888e  .@.U-!.!........
         0100 0024 0222 0024 1101 0018 d51b 8d53  ...$.".$.......S
         c087 9888 fdee 7e85 0a08 add4 626b d61b  ......~.....bk..
         d66e 53a7 6265 7374                      .nS.best

Captured LEAP auth success:

         0802 d500 000c 3043 a907 0007 50ca f417  ......0C....P...
         0007 50ca f417 5067 aaaa 0300 0000 888e  ..P...Pg........
         0100 0004 0313 0004                      ........


Captured LEAP exchange information:
        username:   best
        challenge:  7e46733d63a5fabf
        response:   d51b8d53c0879888fdee7e850a08add4626bd61bd66e53a7
        Attempting to recover last 2 of hash.
        hash bytes: 9537
        Starting dictionary lookups.
        NT hash:    0cb6948805f797bf2a82807973b89537
        password:   test
[root@cyanocorax asleap-1.0]# 
```

Example: Asleap, cracking password “test”

# 802.1x – EAP-TTLS

- Created by Funk and Certicom (Internet draft)
- EAP method 21
- Widely implemented, also Linux support; but no Cisco support
- Supports ANY inner authentication method
  - Any EAP method
  - As well as older methods such as CHAP, PAP, MS-CHAP and MS-CHAPv2

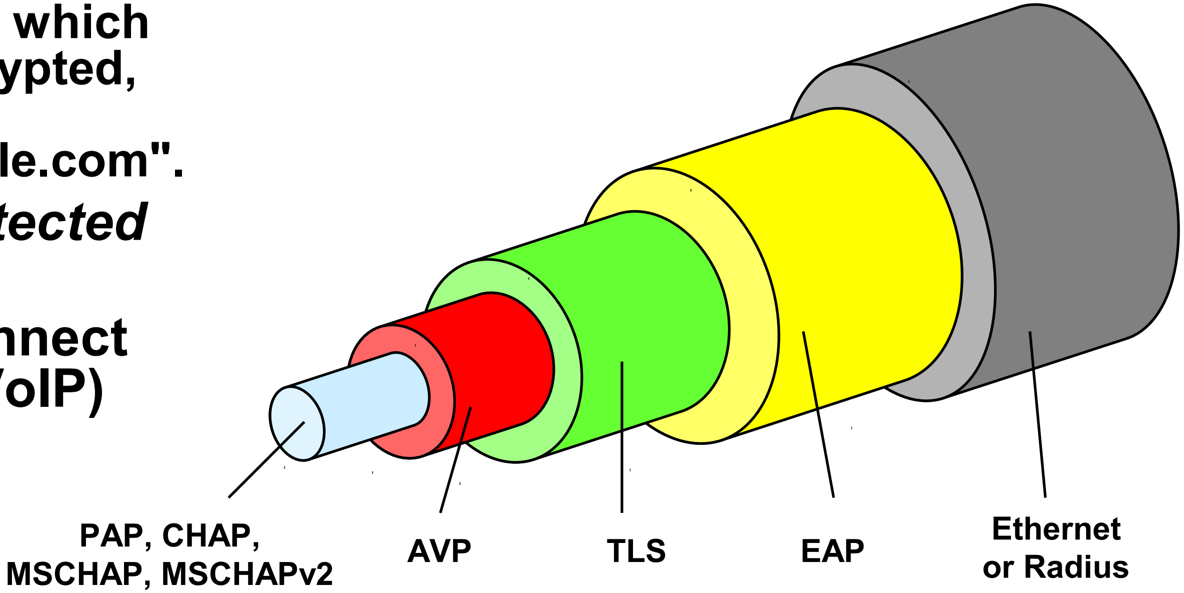
Basic Idea:



# 802.1x – EAP-TTLS

- Radius-like AVPs between client and Server
- Client certificate not required but user has two identities:
  1. A anonymous identity such as "anonymous@example.com" and
  2. The real identity, which is only sent encrypted, such as user342@example.com".
- *Client identity protected by TLS*
- Fast session reconnect (but too slow for VoIP)

Detailed:



# 802.1x – Other EAP Choices

- More than 44 EAP types already defined
  - EAP-AKA: username and password (UMTS systems)
  - EAP-MD5: No dynamic WEP keys, no mutual authentication, dictionary attacks possible
    (EAP method 4)
  - EAP-GTC: Generic Token Card (EAP method 6), no mutual authentication
  - PEAP-GTC: Cisco's PEAP method
  - EAP-SIM: Used for SIM-card based devices (3GPP, also known as EAP-GSM)
  - EAP-SRP: Secure Remote Password
  - …
- EAP-FAST: Successor of LEAP
  - See dedicated section
- PEAP-EAP-TLS
  - Another Microsoft solution similar as EAP-TLS

# EAP Types Overview

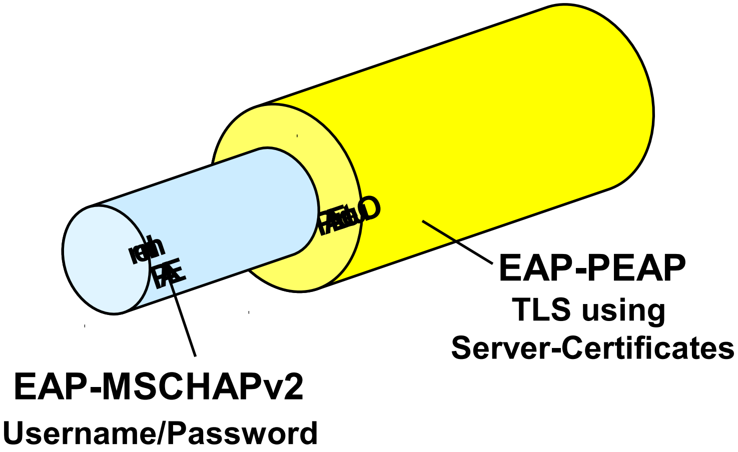
- 1–6 Assigned by RFC
  - 1Identity
  - 2Notification
  - 3Nak (response only)
  - 4MD5-Challenge
  - 5One-Time Password (OTP)
  - 6Generic Token Card (GTC)
- 7-8 Not assigned
- 9 RSA Public Key Authentication
- 10 DSS Unilateral
- 11 KEA
- 12 KEA-VALIDATE
- 13 EAP-TLS
- 14 Defender Token (AXENT)
- 15 RSA Security SecurID EAP
- 16 Arcot Systems EAP
- 17 EAP-Cisco Wireless (LEAP)
- 18 Nokia IP SmartCard authentication
- 19 SRP-SHA1 Part 1
- 20 SRP-SHA1 Part 2
- 21 EAP-TTLS
- 22 Remote Access Service
- 23 UMTS Authentication and Key Agreement
- 24 EAP-3Com Wireless
- 25 PEAP
- 26 MS-EAP-Authentication
- 27 Mutual Authentication w/Key Exchange (MAKE)
- 28 CRYPTOCard
- 29 EAP-MSCHAP-V2
- 30 DynamID
- 31 Rob EAP
- 32 SecurID EAP
- 33 EAP-TLV
- 34 SentriNET
- 35 EAP-Actiontec Wireless
- 36 Cogent Systems Biometrics Authentication EAP
- 37 AirFortress EAP
- 38 EAP-HTTP Digest
- 39 SecureSuite EAP
- 40 DeviceConnect EAP
- 41 EAP-SPEKE
- 42 EAP-MOBAC
- 43 EAP-FAST
- 44–191 Not assigned; can be assigned by IANA on the advice of a designated expert
- 192–253 Reserved; requires standards action
- 254 Expanded types
- 255 Experimental usage

# PEAP

# 802.1x using PEAP

- Created by Cisco and Microsoft
  - Similar to EAP-TTLS
- Open standard
  - EAP method 25
- Since third EAP message is always in clear
  - Client may send a routing realm instead of the user identity to protect the user identity

Basic Idea:



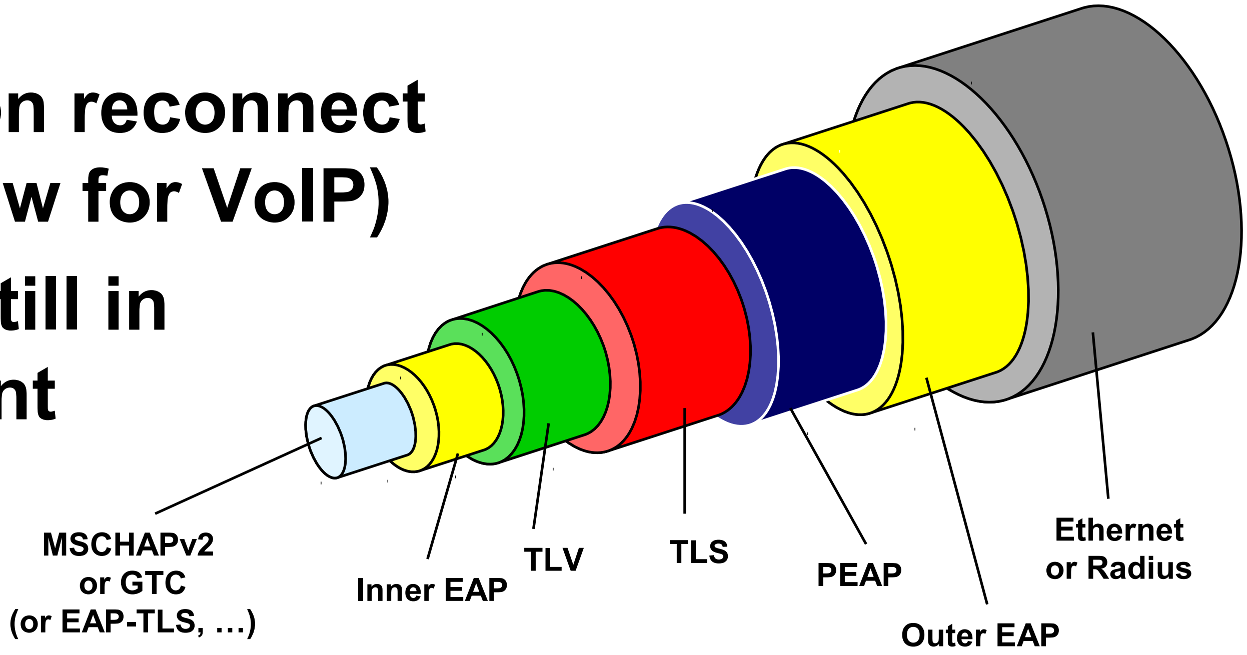
# Version Overview

- PEAPv0
  - Supported since Windows XP SP1
  - Microsoft proposes MS-CHAPv2
    - EAP method 29
- PEAPv1
  - Cisco's proposal: EAP-GTC
    - EAP method 6
- PEAPv2
  - Latest draft
  - Security updates and more features
    - Various cipher-suites supported
    - MITM protection through "crypto-binding"

# PEAP as Pipe Model

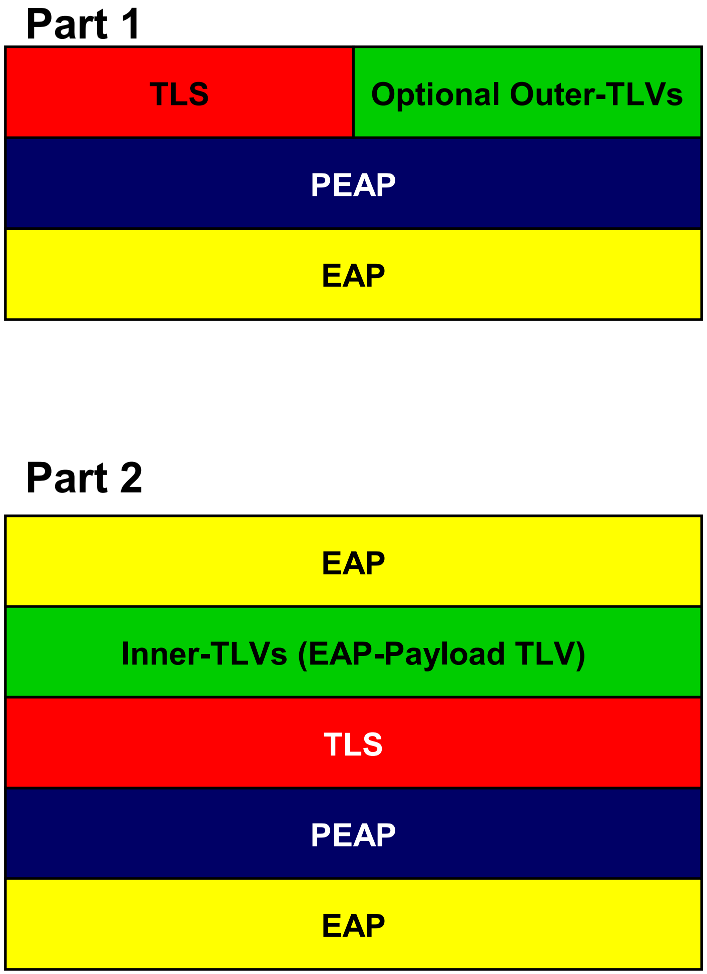
- *Only supports EAP-type authentication*
- Client certificate not required
- Fast session reconnect (but too slow for VoIP)
- Version 2 still in development

PEAP Detailed



# PEAPv2 Layers

- In PEAPv2 Part 1
  - Outer-TLVs are used to help establishing the TLS tunnel, but no Inner-TLVs are used
- In PEAPv2 Part 2
  - TLS records may encapsulate zero or more Inner-TLVs, but no Outer-TLVs
  - EAP packets used within tunneled EAP authentication methods are carried within Inner-TLVs



# PEAPv2: Provisioning of Credentials

- Provisioning inside a server-authenticated TLS tunnel
- Provisioning inside a server-unauthenticated TLS tunnel
  - If TLS tunnel cannot be validated by client (lacking required credentials) the client instead may rely on inner EAP method
  - Although this reduces deployment costs, MITM attacks are possible !
  - An implementation is therefore optional and not recommended

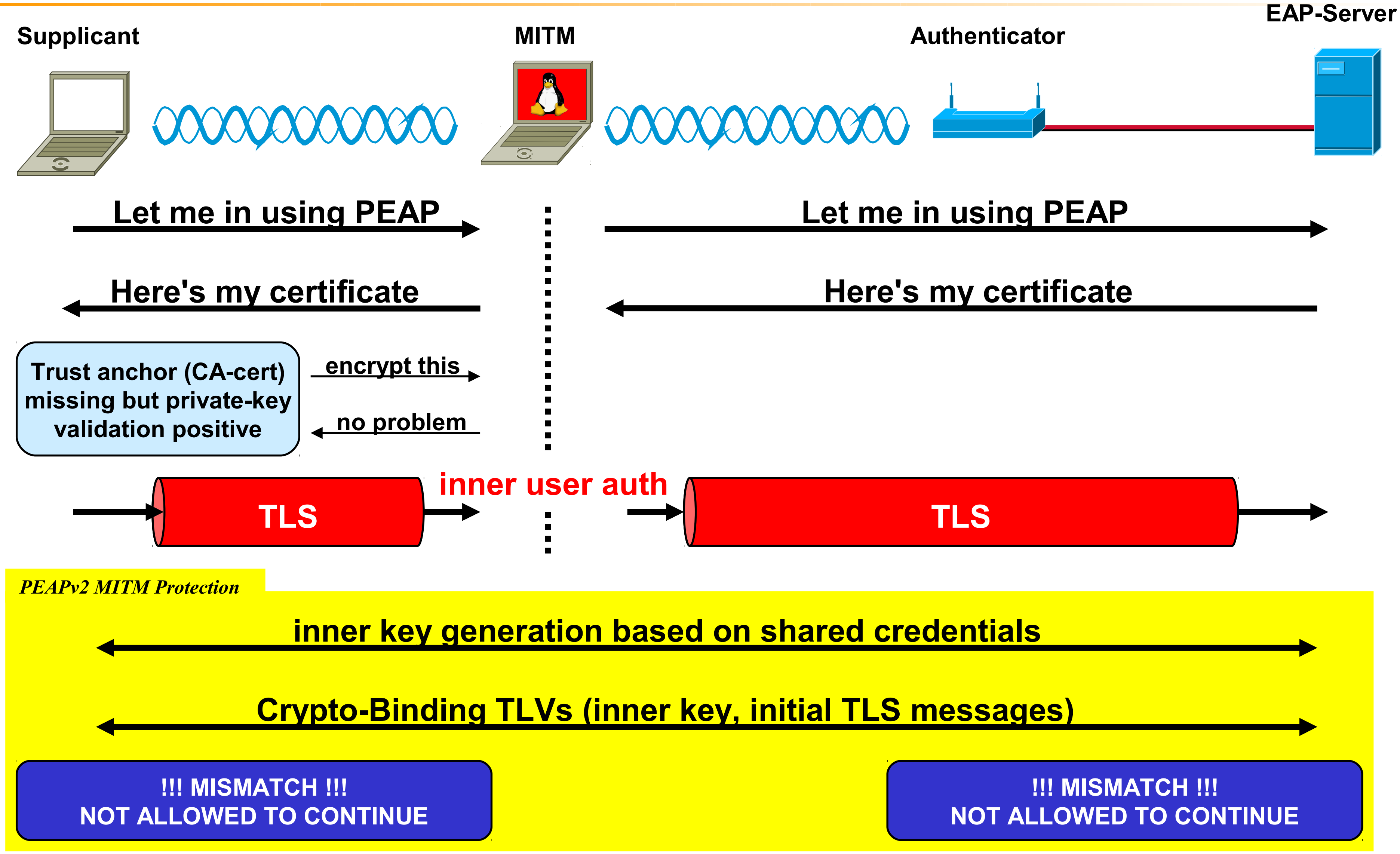
# PEAPv2

- Also other than certificate-based cipher-suites are supported
    - E. g. DH-based
- If certificates are sent by the server
    - The client only verifies whether the server possesses the corresponding private key
    - The client does not need to validate via the trust anchor (CA)

# PEAPv2 – MITM Protection

- A sequence of zero or more inner EAP authentication methods can be negotiated
- Crypto-Binding TLVs must be sent in the PEAP success/failure (Result TLV) messages
  - In a sequence, also after each EAP-method a Crypto-Binding TLV must be sent by both parties
  - The server should not reveal any sensitive data to the client until after the Crypto-Binding TLV has been properly verified !!!

# PEAP: Man-In-The-Middle Attack



EAP-Server

Supplicant

MITM

Authenticator

Let me in using PEAP

Let me in using PEAP

Here's my certificate

Here's my certificate

Trust anchor (CA-cert) missing but private-key validation positive

encrypt this

no problem

inner user auth

TLS

TLS

*PEAPv2 MITM Protection*

inner key generation based on shared credentials

Crypto-Binding TLVs (inner key, initial TLS messages)

!!! MISMATCH !!!
NOT ALLOWED TO CONTINUE

!!! MISMATCH !!!
NOT ALLOWED TO CONTINUE

# Crypto-Binding TLVs

- PEAPv2 derives keys by combining keys from TLS and the inner EAP methods
- The Crypto-Binding TLV calculation includes
  - The first two Outer-TLVs messages sent by both peer and EAP-server
    - (used for TLS tunnel establishment)
  - The EAP-Type (= set to PEAP) sent in the first two messages by both peer and EAP-server

# DoS Attacks

- Theoretically possible if the attacker
  - Can modify unprotected fields in the PEAP packet such as the EAP protocol or PEAP version number
  - Modify protected fields in a packet to cause decode errors

# PEAPv2 – Other Features

- Fast session resumption
  - Using the "sessionID" of the TLS protocol and the Server-Identifier TLV in PEAP
    - Server may send a Server-Identifier TLV  to give client a hint which sessionID should be used (protected by MAC)
  - If too much time elapsed since previous authentication, the server will not allow the continuation
  - The inner authentication may or may not be skipped !!!
- TLS compression must be supported

# PEAPv2 Fragmentation

- A single TLS message may consist of multiple TLS records
  - A single TLS record may be up to 16384 bytes in length
  - A TLS certificate message may in principle be as long as 16 MByte
- Fragmentation needed
  - RADIUS cannot handle such long messages
  - Multilink PPP (MRRU LCP) method supported on Ethernet/802.3
    - But there's no PPP in 802.11 which could negotiate that
  - PEAPv2 own fragmentation support defined
    - DoS attacks (reassembly lockup) can be mitigated to set a maximum size for one group of TLV messages (e. g. 64 KB)

# PEAPv2 Key Derivation

- New keys are derived from TLS master secret to protect the conversation within the PEAPv2 tunnel
  - Since normal TLS keys are used in the handshake they should not be used in a different context
- Combines key material from TLS exchange with key material from inner key generating EAP methods
  - To bind inner authentication mechanisms to TLS tunnel

# Crypto-Binding TLV

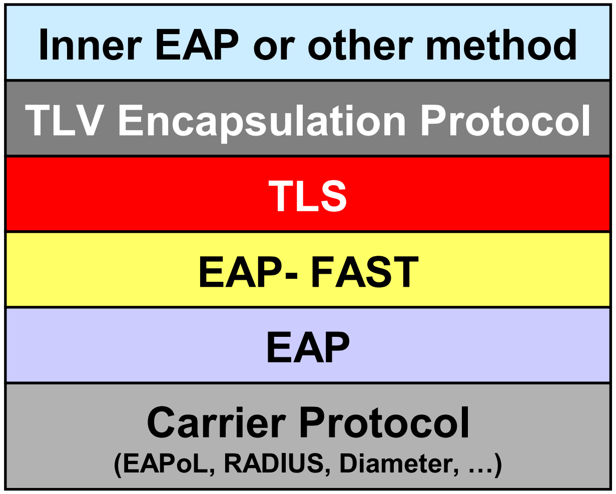
```
 0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |M|R|         TLV Type (12)     |            Length (56)        |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |    Reserved   |     Version   |  Received Ver.    | Sub-Type  |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                                                               |
  ~             Nonce (32 bytes; temporally unique;               ~
  |      used for compound MAC key derivation at each end         |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |                        Compound MAC                           |
  ~ (Computed using the HMAC-SHA1-160 keyed MAC that provides 160 ~
  |  bits of output using the CMK key)                            |
  |                                                               |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- The Crypto-Binding TLV is used prove that both peers participated in the sequence of authentications
  - That is, the TLS session and inner EAP methods that generate keys

# EAP-FAST

# 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



Inner EAP or other method

TLV Encapsulation Protocol

TLS

EAP- FAST

EAP

Carrier Protocol

(EAPoL, RADIUS, Diameter, ...)

# 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

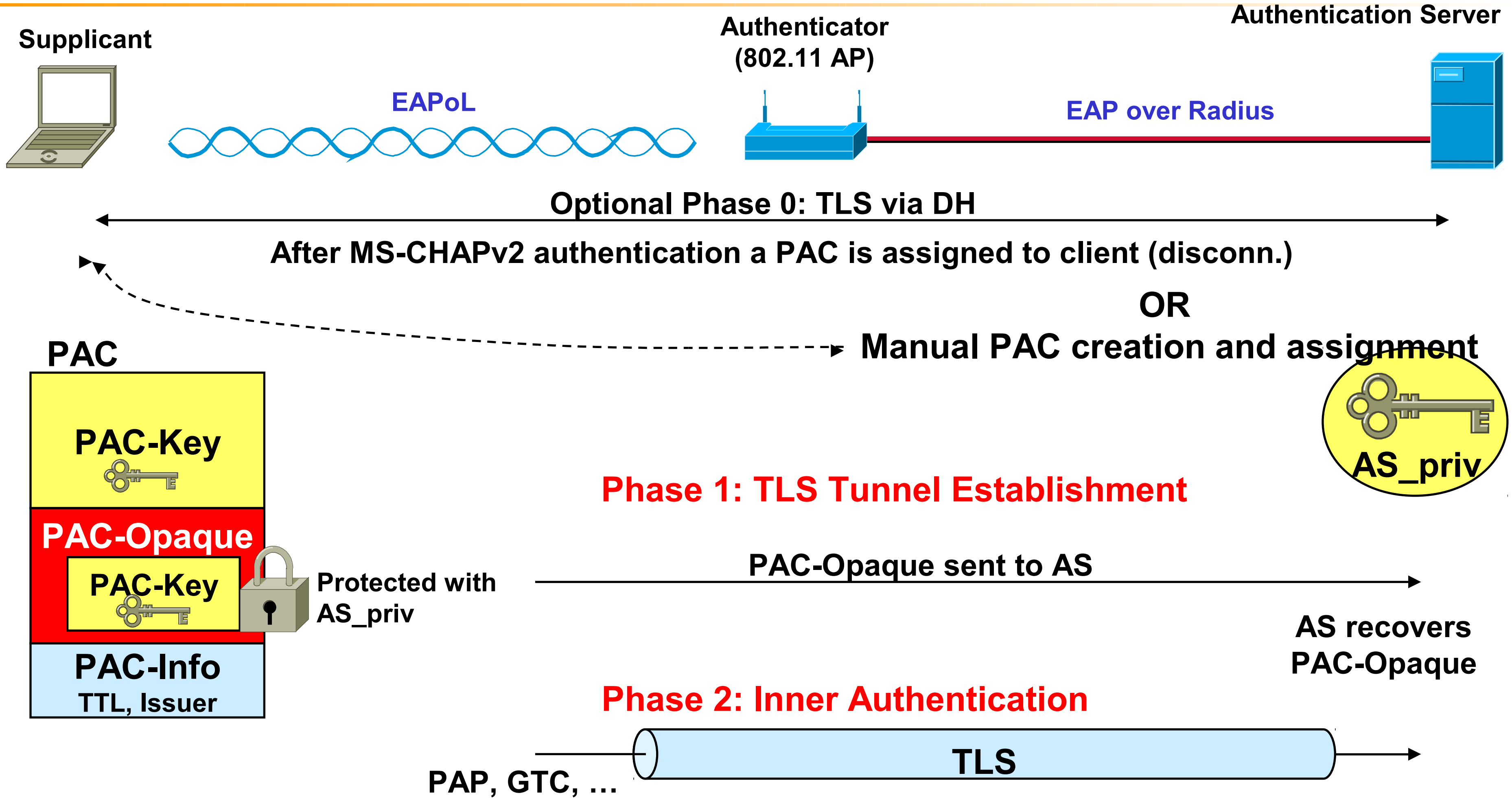
# PAC Components (Detailed)

- 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

# 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

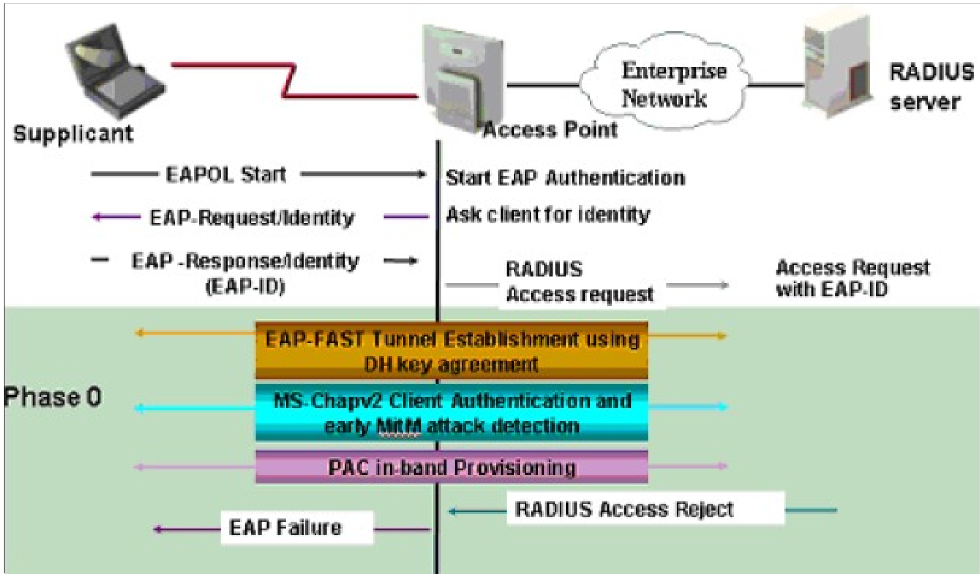


# Note

- 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

# Unauthenticated Phase 0 - Detailed

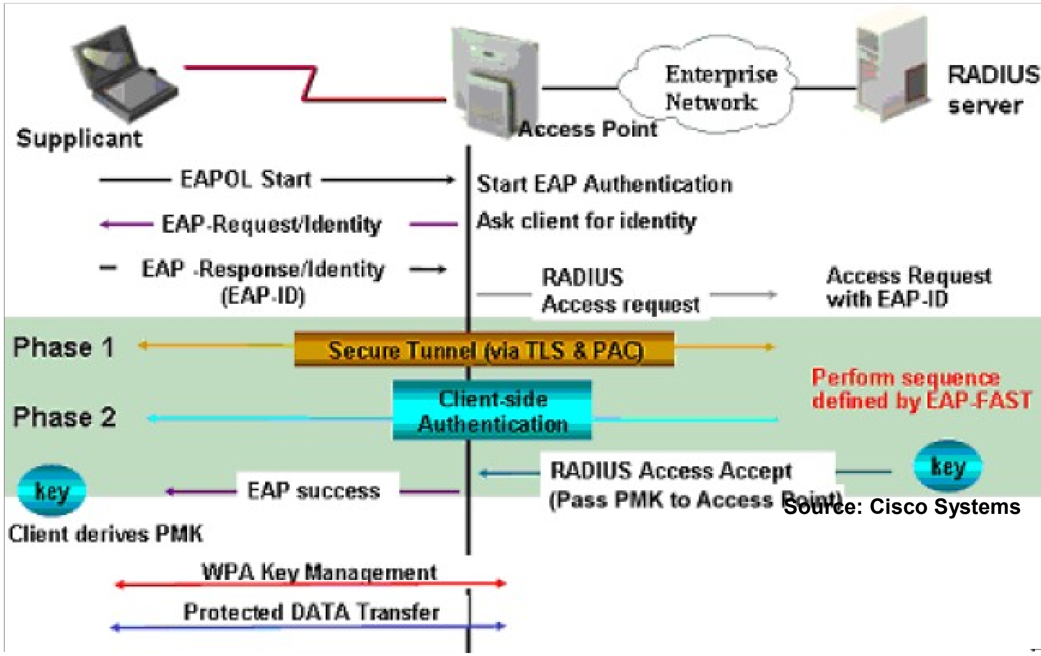
- PAC auto-provisioning using TLS with DH key agreement to establish a secure tunnel
- Additionally, MS-CHAPv2 is used to authenticate the client and to prevent MITM
- After the PAC has been successful provisioned, EAP-FAST is restarted to gain network access
  - Therefore, after a successful PAC provisioning transaction, an EAP *failure* occurs to terminate the EAP-FAST session
  - Afterwards, the newly provisioned PAC can be used to establish an authenticated session



Source: Cisco Systems

# EAP-FAST Phases - Detailed

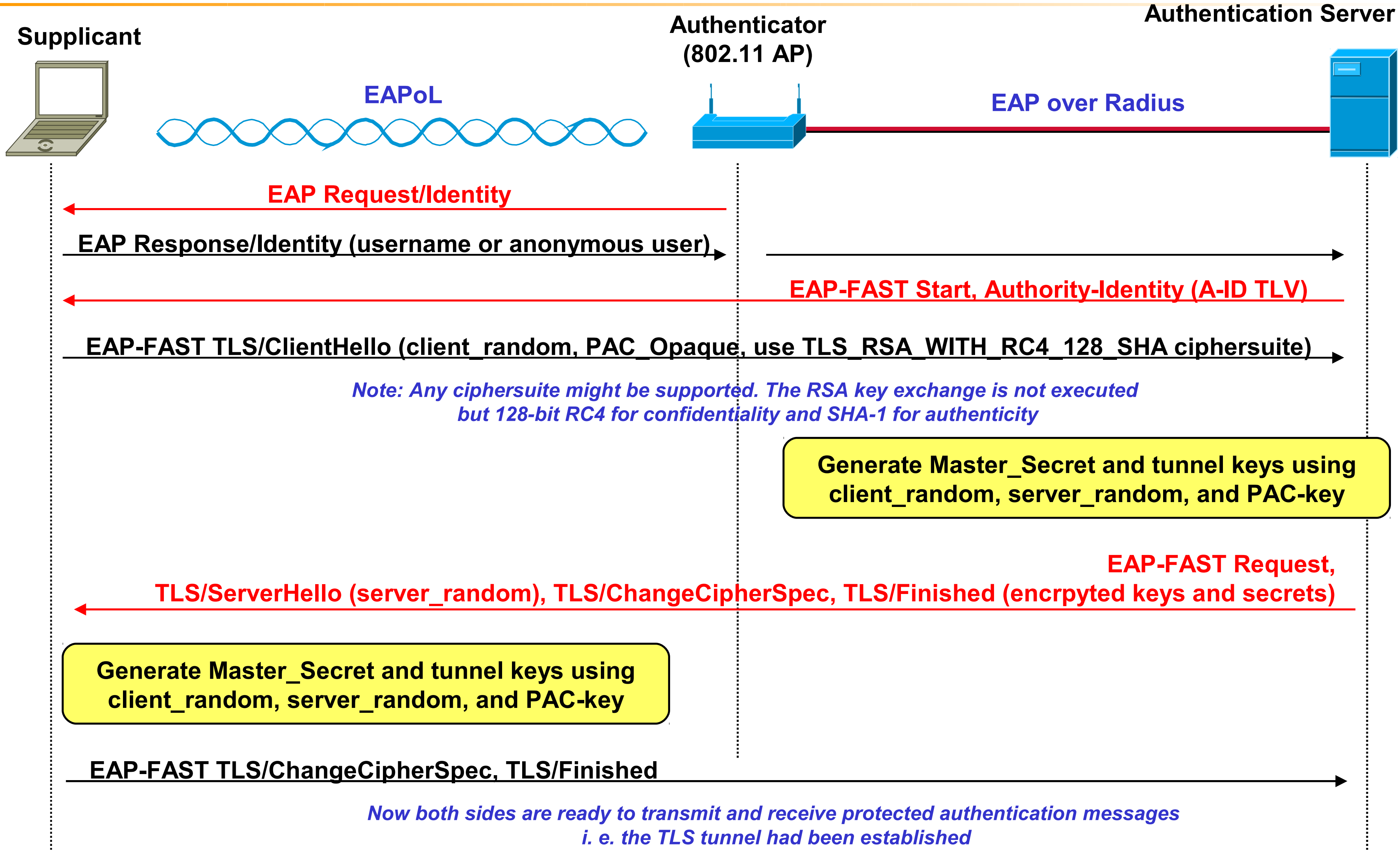
- **Phase 1**
  - **Client sends only the PAC opaque to the server, not the PAC key**
  - **The server decrypts the PAC opaque using its master-key**
    - **Now server and client have the same PAC key**
  - **The PAC key is used to create a TLS tunnel for this client's authentication**
- **Phase 2**
  - **Inside the TLS tunnel, user authentication credentials are passed securely (Phase 2)**
    - **E. g. using EAP-GTC**



Source: Cisco Systems

Source: Cisco Systems

# Phase 1 – Details



Supplicant

Authenticator (802.11 AP)

Authentication Server

EAPoL

EAP over Radius

EAP Request/Identity

EAP Response/Identity (username or anonymous user)

EAP-FAST Start, Authority-Identity (A-ID TLV)

EAP-FAST TLS/ClientHello (client_random, PAC_Opaque, use TLS_RSA_WITH_RC4_128_SHA ciphersuite)

*Note: Any ciphersuite might be supported. The RSA key exchange is not executed but 128-bit RC4 for confidentiality and SHA-1 for authenticity*

Generate Master_Secret and tunnel keys using client_random, server_random, and PAC-key

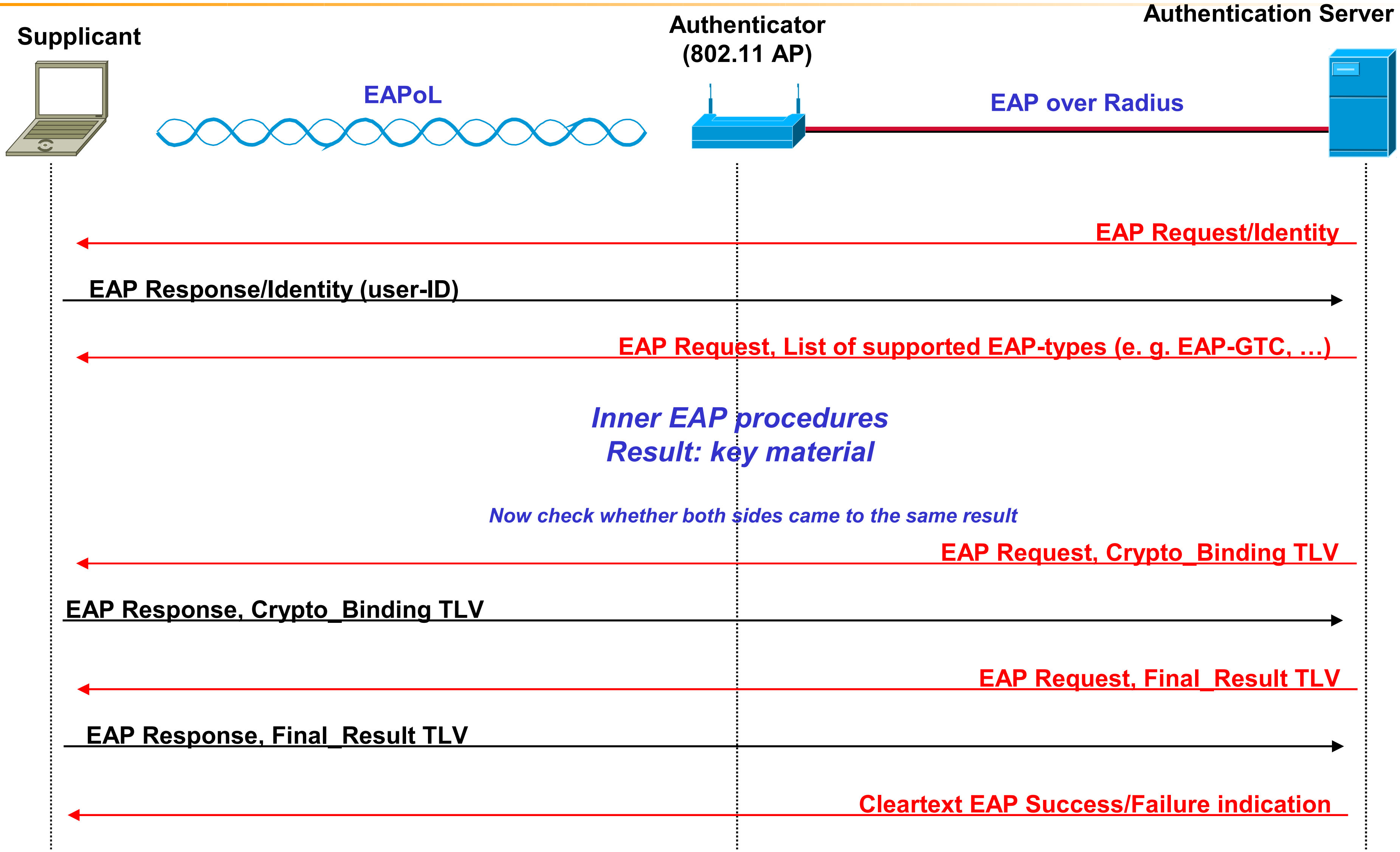
EAP-FAST Request,
TLS/ServerHello (server_random), TLS/ChangeCipherSpec, TLS/Finished (encrpyted keys and secrets)

Generate Master_Secret and tunnel keys using client_random, server_random, and PAC-key

EAP-FAST TLS/ChangeCipherSpec, TLS/Finished

*Now both sides are ready to transmit and receive protected authentication messages i. e. the TLS tunnel had been established*

# Phase 2 – Details



EAP Request/Identity

EAP Response/Identity (user-ID)

EAP Request, List of supported EAP-types (e. g. EAP-GTC, …)

*Inner EAP procedures*
*Result: key material*

*Now check whether both sides came to the same result*

EAP Request, Crypto_Binding TLV

EAP Response, Crypto_Binding TLV

EAP Request, Final_Result TLV

EAP Response, Final_Result TLV

Cleartext EAP Success/Failure indication

# 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
  - Also supports separate machine authentication
- Seamless migration from LEAP to EAP-FAST possible
  - Similar AP settings
  - ACU reconfiguration via ACAT
- WPA is also supported

# WPA and WPA2

# Introduction

- 802.1x alone does not (need to) provide key management
  - Often 802.1x is simply combined with WEP
  - Even 802.1x with TKIP would always start with same base key
- Basic Idea of WPA:
  - Strong per-user, per-session, per-packet keying (TKIP and MIC)
  - Use 802.1x and dynamical transient key management
  - Alternatively pre-shared keys (SOHO apps.) instead of 802.1x
- WPA starts with a security capability negotiation
  - Therefore cipher suites must be configured on AP
  - APs advertises capabilities in beacon and in probe-response frames
    - "Cipher Suite" = Auth. Method + Encryption Method
  - Client can select the desired method during association request

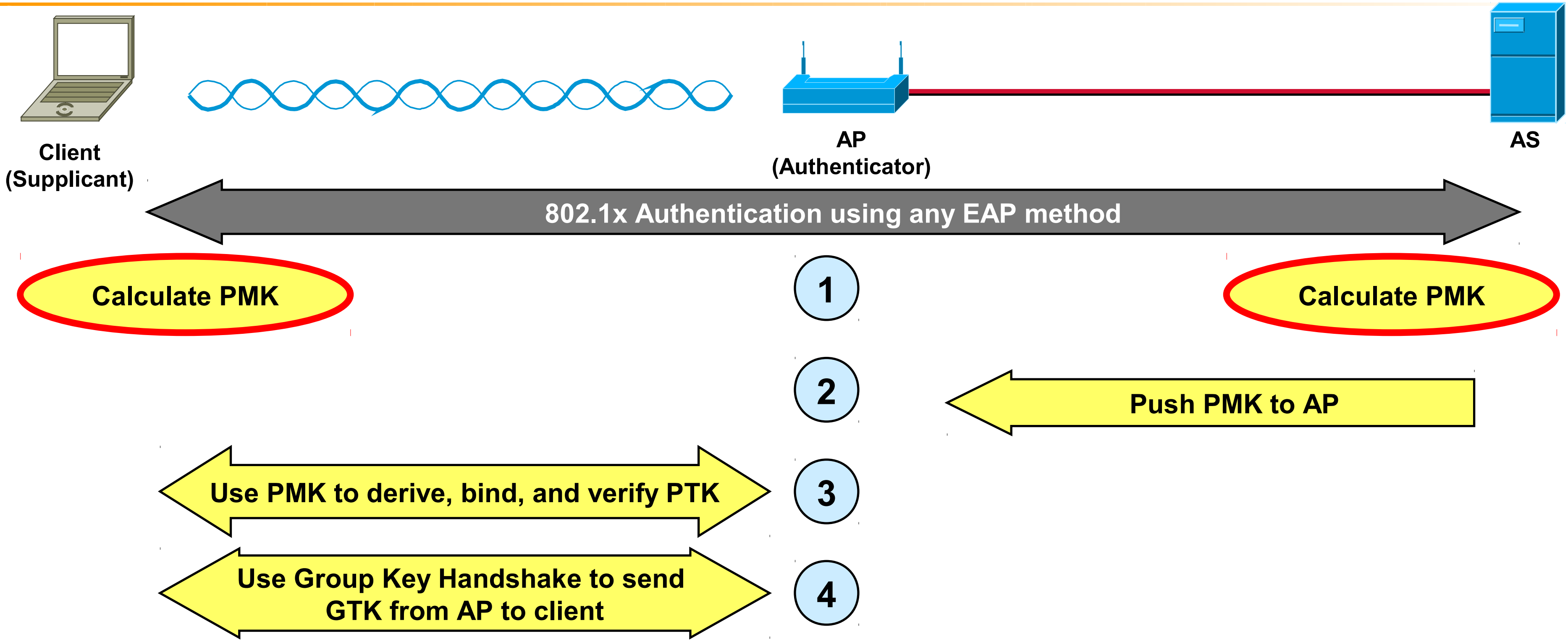
# WPA/WPA-2

- Certified EAP Methods
  - EAP-TLS (originally the only one)
  - EAP-TTLS/MSCHAPv2
  - PEAPv0/EAP-MSCHAPv2
  - PEAPv1/EAP-GTC
  - EAP-SIM
- Native OS support
  - Windows XP with Service Pack 2 and WPA2 patch
  - No support for Win2k
  - Linux: *wpasupplicant* (large feature set)

# WPA Concepts

- 1) Pairwise Master Key (PMK) is negotiated between client and AS
  - Based on 802.1x credentials or based on a PSK in home environments
  - PMK is designed to last the entire session
  - Should be exposed as little as possible (therefore PTK needed)
- 2) PMK is pushed from AS to AP
  - Via RADIUS-Access-Accept message
- 3) AP generates Pairwise Transient Key (PTK)
  - Negotiated via Four-Way Handshake to client
  - PTK= HASH (PMK, AP_nonce, STA_nonce, AP_MAC, STA_MAC)
  - From PTK, other working keys are generated (KCK, KEK, TK)
- 4) AP also derives a Group Temporal Key (GTK)
  - To decrypt multicast and broadcast traffic
  - Must be the same on all clients (!)
  - Need to be updated periodically (e. g. when a device leaves the network)
  - AP sends new GTK to each client, encrypted with client's PTK
  - Each client must acknowledges the new GTK

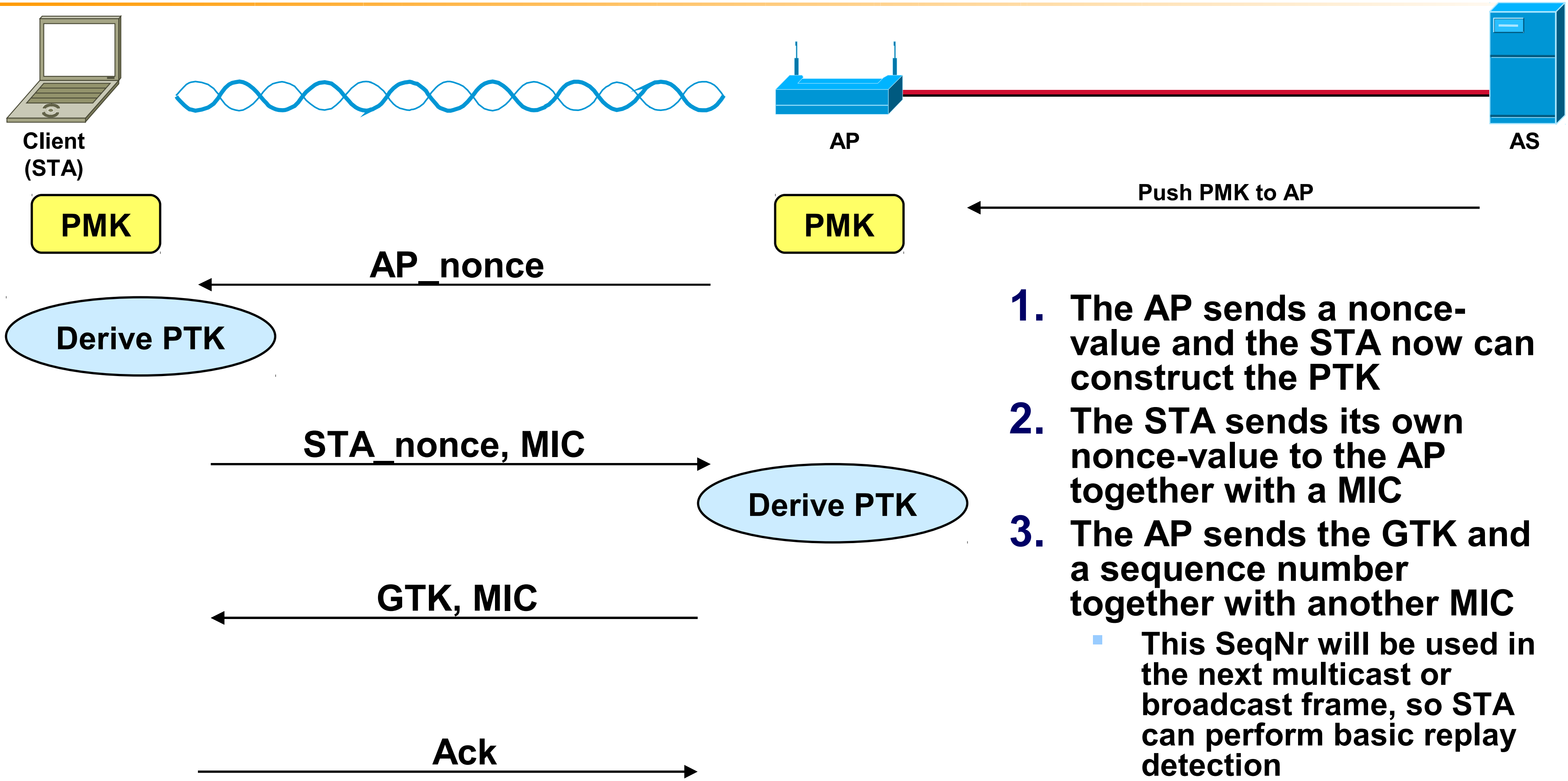
# The Basic Steps



Client
(Supplicant)
AP
(Authenticator)
AS
802.1x Authentication using any EAP method
Calculate PMK
1
Calculate PMK
2
Push PMK to AP
Use PMK to derive, bind, and verify PTK
3
Use Group Key Handshake to send
GTK from AP to client
4

- PMK is derived from the master key of the preceding 802.1x negotiations
- Four WPA (main-) steps are performed after 802.1x authentication
- Each step of this procedure is protected by dedicated transient (temporary) keys

# WPA – Basic Handshake (Simplified)

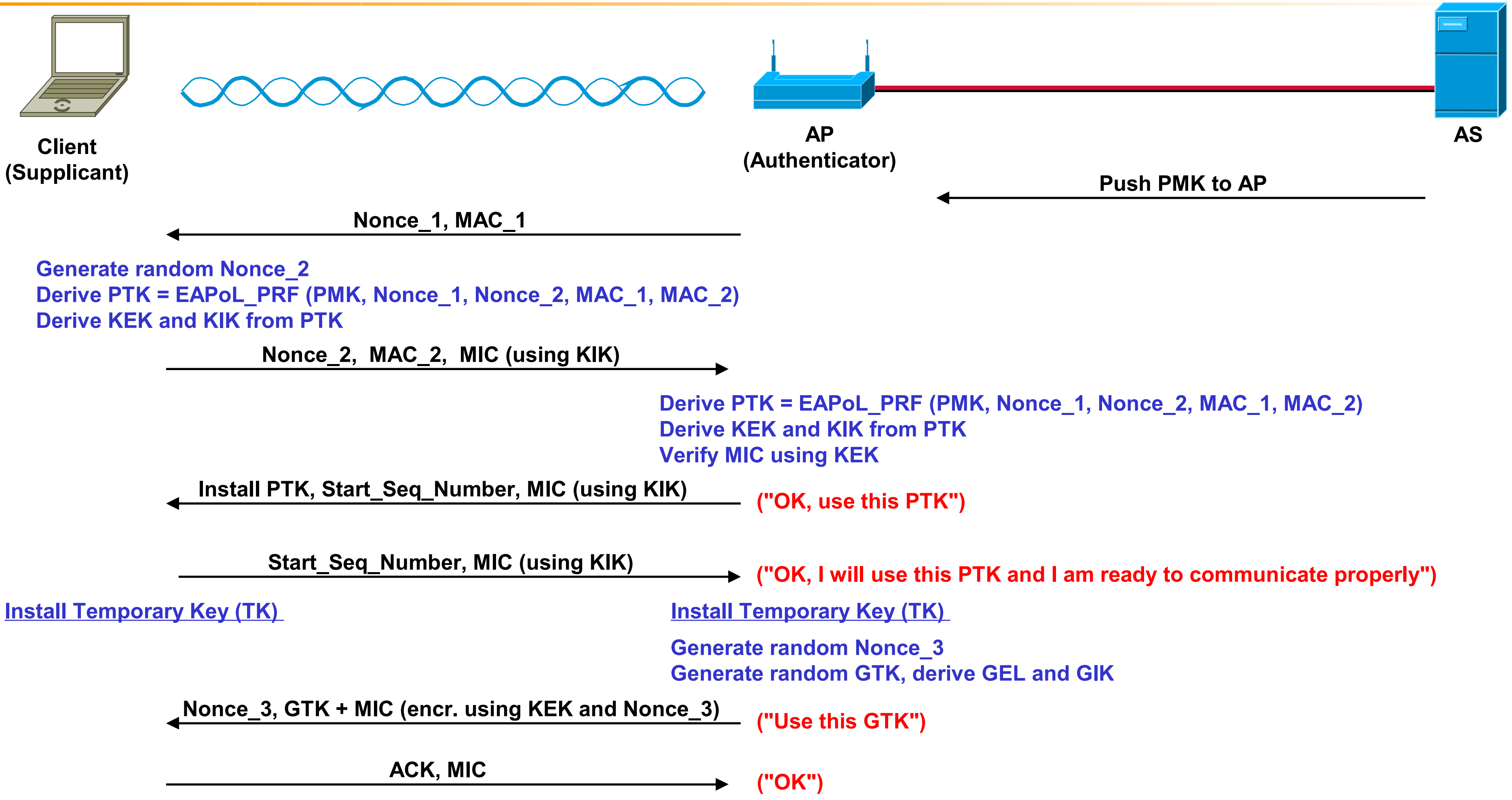


1. The AP sends a nonce-value and the STA now can construct the PTK
2. The STA sends its own nonce-value to the AP together with a MIC
3. The AP sends the GTK and a sequence number together with another MIC
   - This SeqNr will be used in the next multicast or broadcast frame, so STA can perform basic replay detection
1. The STA sends a confirmation to the AP

# WPA Details – Transient Keys

- The PTK (256 bit) is the basis to derive additional transient keys
  - Data Encryption Key (128 bit)
    - For unicast frames
    - Aka Temporal Key (TK)
  - Data Integrity Key (128 bit)
    - For unicast MIC
  - Key Encryption Key (KEK, 128 bit)
    - To encrypt EAPoL key messages
  - Key Integrity Key (KIK, 128 bit)
    - To calculate the MIC for EAPoL key messages
- The GTK (256 bit) is the basis to derive
  - A Group Encryption Key (GEK)
  - A Group Integrity Key (GIK)

# (WPA – Detailed)



Client (Supplicant)

AP (Authenticator)

AS

Push PMK to AP

Nonce_1, MAC_1

Generate random Nonce_2
Derive PTK = EAPoL_PRF (PMK, Nonce_1, Nonce_2, MAC_1, MAC_2)
Derive KEK and KIK from PTK

Nonce_2, MAC_2, MIC (using KIK)

Derive PTK = EAPoL_PRF (PMK, Nonce_1, Nonce_2, MAC_1, MAC_2)
Derive KEK and KIK from PTK
Verify MIC using KEK

Install PTK, Start_Seq_Number, MIC (using KIK)

("OK, use this PTK")

Start_Seq_Number, MIC (using KIK)

("OK, I will use this PTK and I am ready to communicate properly")

Install Temporary Key (TK)

Install Temporary Key (TK)

Generate random Nonce_3
Generate random GTK, derive GEL and GIK

Nonce_3, GTK + MIC (encr. using KEK and Nonce_3)

("Use this GTK")

ACK, MIC

("OK")

- All WPA procedure messages are of type "EAPoL Key Messages"
- Temporary Key (TK) consists of (256-n) bits of the PTK, depending on cipher used
- Same Group Transient Key (GTK) is assigned to all clients within VLAN

# GTK Issues

- GTK is either
  - A pseudo-random number chosen by AP
  - The first PTK that the AP uses
- GTK Usage
  - Cannot be used with sequence numbers because it is used for ALL clients
    - Distant clients might overhear some frames
  - So management and broadcast frames are encrypted via WEP only
    - Broadcast key rotation recommended

# WPA-2: PKC

- WPA2 mandates both TKIP and AES capability
  - TKIP is used by the network if at least one client supports TKIP only
- PMK Proactive Key Caching (PKC) support
  - AP caches credentials 1 hour to allow fast reconnect

# WPA-2: Pre-Authentication

- Pre-authentication support
  - Allows a client to pre-authenticate with the AP toward which it is moving
  - But still maintains a connection to the AP it's moving away from
- Note that pre-authentication is done through the AP to which the client is currently assoicated!
- Roaming times below 100 ms

# WPA-PSK (1)

- ONLY useful for home WLANs
- Relies on Pre-Shared Key (PSK) only
- No AAA server needed
- PMK is a 4096-times hash of:
  - Passphrase (8-63 chars or 64 hex digits)
  - SSID and SSID-length
  - Nonces

# WPA-PSK (2)

- 2003: Robert Moskowitz published an effective dictionary attack against WPA-PSK
- Passphrase should be more than 20 characters !!!
- Attack Tools: CoWPAtty, KisMAC, WPA Cracker, …