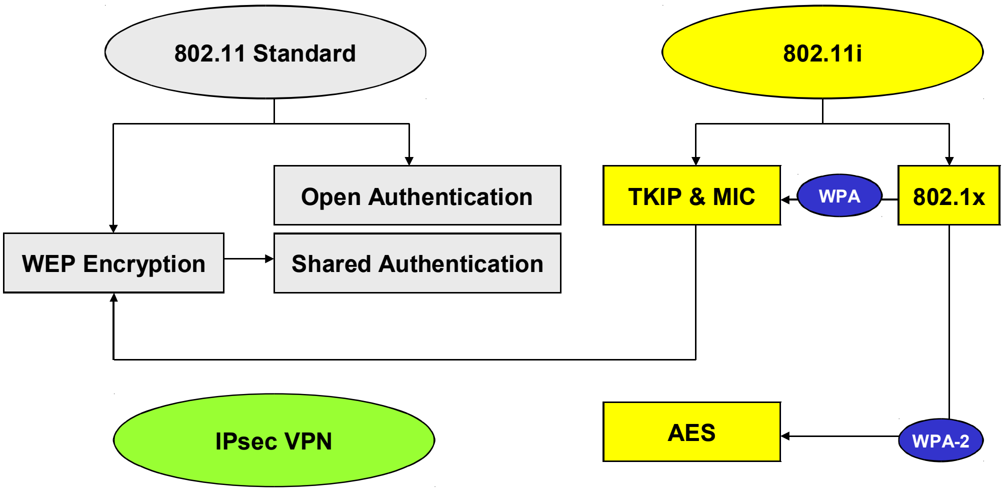
WLAN

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

Content

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

This subchapter provides an introduction into WEP, the basis of the 802.11 original and only method for encryption, authentication and integrity protection.

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)

Compared to all other physical communication media, the wireless realm is the **best-of-choice medium for attackers** and hackers. The main reason for this is, that there are no wires an attacker needs to attach to. Moreover, the attacker can hide in another building or in a car, more than 100 meters outside the building (if he/she has a good antenna).

Since there are no wires it is not possible to protect the physical media from interferences or jamming, therefore **DoS attacks** are critical. An attacker could even destroy the sensitive receiving devices by jamming at very high power levels.

Additionally, there are other problems, caused by the 802.11 design itself:

- The standard encryption, integrity and authentication method has serious **design flaws**.
- There is no means to **manage user credentials** in a central way, which leads to bad practical security designs.
- The standard security concept is based on **device-bound secrets**, therefore hardware theft opens security holes for that network.
- The standard security concept does not allow to authenticate the infrastructure devices, therefore so-called "**rogue access points**" can be installed by attackers.
- Proprietary security enhancements caused an **interoperability problem** for several years.
- Dozens of **cracker tools** are available on the Web.

And finally, the WLAN **security awareness** only became widespread in the last year and still too many WLAN networks are poorly secured or not secured at all. In 2002/2003, almost two-third of all WLAN networks were unprotected.

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} !!!**

The security of stream ciphers depends 1) on the pseudo-randomness of the keystream they produce, and 2) of the implementation which must guarantee that each keystream is only used once! Since encryption and decryption is the same operation (XOR), if two plaintexts are encrypted with the same keystream, cryptanalysis is typically simple (for example, assume that one plaintext is known).

A stream cipher can be as secure as block cipher of comparable key length but typically stream ciphers are much faster and use far less code. For example, if 3DES can produce 3 Mbit/s on a Pentium II, then RC4 could achieve 45 Mbit/s, which is 15-times faster!

The RC4 algorithm had been kept as a trade secret by RSA Security, but in September 1994 the code was anonymously posted in the Cypherpunks mailing list.

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.

Possible key lengths range from 1 to 256 bytes (i. e. 8 to 2048 bits).

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(!)

The Wired Equivalent Privacy (WEP) algorithm should provide a nearly-wired privacy look-and-feel, as its name suggests.

WEP uses the **RC4** PRNG algorithm from RSA Data Security Inc. RC4 is a stream cipher, a well studied algorithm, which expands a key into an infinite pseudorandom sequence.

This RC4 key consists of a **40 bit or 104 bit secret key** and a **24 bit Initialization Vector (IV).**

Note: The 40- or 104-bit WEP key is used as the base key for each packet. When combined with the 24-bit initialization vector, it is sometimes called the "**WEP seed**". Therefore WEP seeds are made of 64 or 128 bits in total and many manufacturers refer to the 104-bit WEP keys as 128-bit keys for this reason.

Unfortunately the IEEE 802.11 standard does not specify methods how to distribute the WEP keys to infrastructure and client devices.

Typically, most vendors allow to specify **up to four WEP keys** which can be dynamically chosen in order to confuse attackers.

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**

Both the visible initialization vector and the shared secret WEP key are used by the RC4 algorithm to produce a pseudo-random **keystream** for encryption and decryption.

This keystream is mixed with the payload using the **XOR operation**. In principle the RC4 encryption is very secure—if there were no severe design flaws.

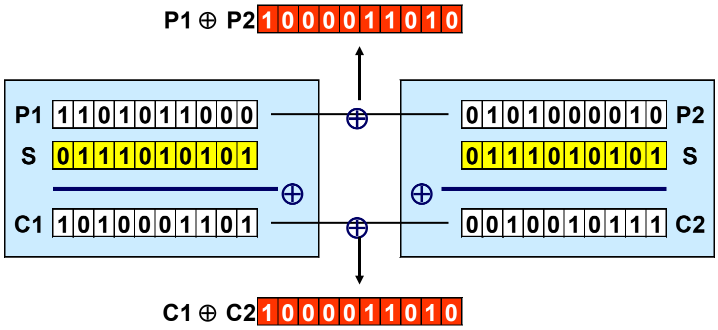
The weaknesses within WEP were first exposed by researchers from Intel, the University of California at Berkeley, and the University of Maryland. The most damning report came from Fluhrer, Mantin, and Shamir, which outlined a passive attack that Stubblefield, Ioanndis, and Rubin at AT&T Labs and Rice University implemented by capturing a hidden WEP key based on the attacks proposed in the Shamir et al. paper (aka **Fluhrer et. al.** paper). This attack took just hours to implement.

Ron Rivest, inventor of the RC4 algorithm, recommends that

> "Users consider strengthening the key scheduling algorithm by preprocessing the base key and any counter or initialization vector by passing them through a hash function such as MD5. Alternatively, weaknesses in the key scheduling algorithm can be prevented by discarding the first 256 output bytes of the pseudo-random generator before beginning encryption. Either or both of these techniques suffice to defeat the [Fluhrer, Martin, and Shamir] attacks on WEP."

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!**



(C) Herbert Haas 2010/02/15 11

Although RC4 is a very good algorithm, its application with WEP reveals some remarkable security flaws. WEP is insecure when the **same keystream is used more than once**—the key length and the random properties of the keystream do not matter at all!

This is because the **XOR operation eliminates two identical terms**. That is, if an attacker sniffed Ciphertext C1 and Ciphertext C2, which had been produced by the same keystream S, then actually the following operations were made by the WEP algorithm:

C1=S ⊕ P1 and C2=S ⊕ P2.

Hence C1 ⊕ C2 cancels out S and equals P1 ⊕ P2. Thus, if Plaintext P1 is known, P2 can be easily calculated!

Note: This attack method also works for a subset of these "vectors": If a part of P1 is known, then a congruent part of P2 can be calculated.

Knowledge of parts of the plaintext message can enable **statistical attacks** to recover all plaintexts. These statistical attacks become increasingly practical as more ciphertexts that use the same key stream are known. Once one of the plaintexts becomes known, it is trivial to recover all of the others.

Although most 802.11 equipment is designed to disregard encrypted content for which it does not have the key, it is relatively simple to change the configuration of the drivers. Active attacks, which requires transmission seems to be more difficult, yet not impossible. Many 802.11 products come with programmable firmware, which had been reverse-engineered and modified to provide the ability to inject traffic to attackers.

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!

Because of the XOR properties it is crucial to continuously change the key that makes up the particular keystream—ideally for each packet sent! The key is made up of the shared secret and the IV, and the latter was intended to assure collision protection. But actually, **the standard does not specify how to change the IV.** There is no strict requirement to change IVs at all!

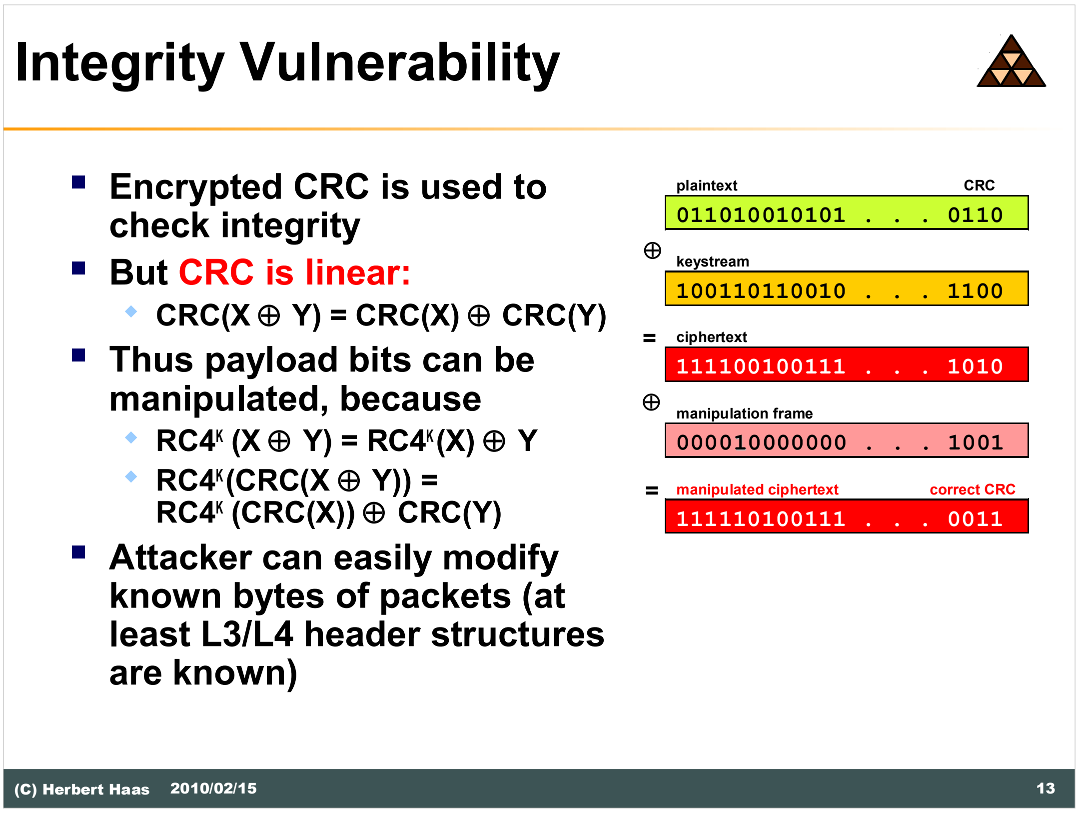
Example of an attack duration:

> A busy access point, which constantly sends 1500 byte packets at 11Mbps, will exhaust the space of IVs after 1500*8/(11*10^6)*2^24 = ~18000 seconds, or 5 hours. This allows an attacker to collect two Ciphertexts that are encrypted with the same key stream and perform statistical attacks to recover the plaintext.

Now it is clear, that the shared **key length** do not affect this sort of attack at all (also see Jesse Walker's "Unsafe at any key length" paper). If P1 is known then P2 is immediately available. Much of network traffic contains predictable information, but it is much easier when three or more packets collide. Certain devices on the market utilize the IV in a simply **predictable** way, for instance by incrementing by one for each packet. Furthermore, the IV value is reset at each startup.

> One New York computer security consultant who was quoted in the Wall Street Journal article says he was able to access the computer network of his client, a major financial services firm on Wall Street, while sitting on a bench across the street.

Common wireless sniffing tools are **WEPcrack** and **AirSnort.**



Furthermore, WEP is also used to protect the integrity of a frame in combination with the **CRC**. But the CRC is a **linear operation** and can therefore be **additively decomposed.**

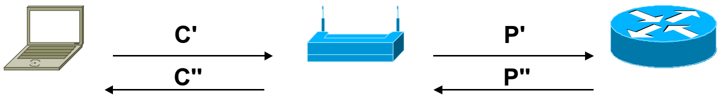
Because of this property, an attacker could XOR a plaintext X with another plaintext Y for manipulation purposes and only has to calculate CRC(X) XOR CRC(Y) to get CRC(X XOR Y). Because of the linearity, this operation can also be successfully applied even when the CRC is RC4-encrypted!

Thus the "Integrity check" does not prevent packet modification, and an attacker can **easily flip bits in packets**, modify active streams, or bypass access control.

Even partial knowledge of the packet is sufficient if the attacker wants only to modify the known portion.

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!

(C) Herbert Haas 2010/02/15 15

The Arbaugh Attack

Here is a more detailed example to understand the Arbaugh attack:

1. Find an initial keystream S of size n. For example look for DHCP-Discover messages, which have a fixed size and a broadcast MAC destination address. The known plaintext of the DHCP-Discover message consists of a source IP address of 0.0.0.0, a broadcast destination IP address 255.255.255.255, and some other fixed header information. This method reveals 24 bytes of keystream, that is n = 24.
2. Create a message M of size n - 3, that is 21 bytes in our case. For example an ARP request or an ICMP packet.
3. Create the ICV of the message M and append three bytes of it to the message, resulting in a plaintext P .
4. XOR the known keystream to the plaintext: C = P XOR S.
5. Instead of the true fourth byte of the ICV append a test byte Bi to the ciphertext C. For example the first test byte could be B0 = 0x00. The resulting ciphertext packet is Ci .
6. Send Ci to the AP. If the last byte Bi (i. e. the fourth byte of the ICV) was correctly encrypted then the AP accepts the packet and the network will send a response. If Bi was wrongly encrypted then the AP will discard the packet silently. Next try B1 = 0x01, B2 = 0x02, ... ,B255 = 0xFF. On average after 128 trials Bi is found.
7. Since the whole ICV is known as plaintext, calculate the unknown keystream-byte S25 = Bi XOR ICV4 . Remember that Bi = ICV4 XOR S25 .

Practically one could create an ICMP echo request of increasing length. If the frame has been correctly encrypted then there will be an ICMP echo reply. (Remember that the payload of an ICMP packet may have arbitrary length.

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**

The presented WEP attacks only belong to the most simple one. Here is a summary of the most practical attack methods.

Keystream reuse attack:

This already described method is typically combined with dictionary-building and statistical analysis. Finally the attacker has created a large dictionary containing all keystreams possible with the used WEP keys and then he/she can perform real-time decryption of all traffic.

Bit flipping attacks:

The attacker could make guesses about the headers of a packet, which contains typically a lot of redundancy that is predictable. In particular, all that is necessary to guess is the destination IP address. Now the attacker can flip appropriate bits to transform the destination IP address to send the packet to another machine, which is in his own realm. Most wireless networks are connected to the Internet and the APs will decrypt each packet that is destined to a wired destination. This is also called a redirection attack.

If a guess can be made about the TCP headers of the packet, the attacker could change the destination port to be port 80, which will allow it to be forwarded through most firewalls. Note that the IP checksum can be easily spoofed and the TCP checksum is disregarded by the network.

Changing an IP address is relatively simple. Assume the high and low 16-bit words of the original IP address are IP_H1 and IP_L1, and should be changed to IP_H2 and IP_L2. If CRC1 is the original IP checksum, then CRC2 = CRC1 + IP_H2 +IP_L2 – IP_H1 – IP_L1 in one's complement math.

If the attacker knows CRC1 by some means, he then figures out CRC2 as above and computes CRC1 XOR CRC2 to get to the final checksum. Another way is to make guesses about the IP address and see if they work. The TCP reaction attack works by seeing what the reaction of the system is to forgeries. A correctly guessed IP will be accepted by the system, while a bad one causes the packet to be dropped into the bit bucket. This only works on TCP packets, because the attacker needs the ACKs that TCP sends (the TCP ACK packet is of a standard size) when the TCP checksum is correct.

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

Fluhrer et. al. attack:

Some IV values reveal information about key state, thus the shared keys can be recovered after several million packets. In the RC4 algorithm the Key Scheduling Algorithm (KSA) creates an IV based on the base key. A flaw in the WEP implementation of RC4 allows "weak" IVs to be generated. The RC4 key scheduling is insufficient: the beginning of the pseudorandom stream should be skipped.

The **KoreK Attack** was first implemented in the tool "ChopChop" and is now part in nearly all WEP cracking tools, such as aircrack or airsnort.

Also the **Arbaugh Attack** is an acceleration tool and therefore part in many modern WEP cracking tools.

Interim Solutions: TKIP and MIC

Content

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

This subchapter provides an introduction into 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

Task Group i (TGi) was formed in March 2001 as a split from the MAC Enhancements Task Group (TGe). Its charge was to "enhance the 802.11 Media Access Control (MAC) to enhance security and authentication mechanisms." TGi finished work on the 802.11i standard, and it has been approved.

802.11i defines two WLAN network types: Transition Security Network (TSN) and Robust Security Network (RSN). RSNs only allow devices which support TKIP/Michael and CCMP. TSNs support both RSN devices and legacy pre-RSN, i. e. WEP devices. The drawback with RSN is that broadcast packets have to be transmitted with the weakest common denominator security method. If there is a device using WEP in a TSN network, it weakens the security of broadcast traffic for all the devices. RSN is definitely preferred, and getting all networks to use CCMP exclusively is the long term goal.

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)

(C) Herbert Haas 2010/02/15 20

Recently, the **IEEE 802.11i** Security Task Group released two "informative texts" providing WEP hardening: MIC and TKIP. The IEEE 802.11 Task Group "i" is working on standardizing WLAN encryption improvements. Two new network types, called Transition Security Network (TSN) and Robust Security Network (RSN) had been defined.

The Temporal Key Integrity Protocol (TKIP, initially referred to as WEP2) is an interim solution (as part of TSN) that fixes the key reuse problem of WEP.

TKIP is a compromise on strong security and possibility to use existing hardware. Still uses RC4 but per-packet keys plus replay protection through a keyed packet authentication mechanism (Michael MIC).

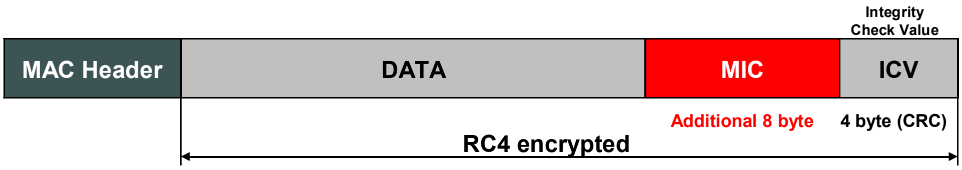
TKIP begins with a 128 bit "temporal key" shared among clients and access points. TKIP combines the temporal key with the client's MAC address and then adds a 6-byte IV to produce the key that will encrypt the data. Thus each station uses different key streams for encryption. TKIP changes keys every 10,000 packets, using a dynamic distribution method.

The IEEE plans to use the **Advanced Encryption Standard (AES)** instead of RC4 for TKIP in the long run (RSN), combined with Counter Mode - Cipher Block Chaining - Message Authentication Code (CBC MAC) to provide strong integrity and message authentication. Also the term "Wireless Robust Authenticated Protocol" (WRAP) is sometimes used synonymously for this concept.

The **Wi-Fi** specified TKIP and MIC as mandatory features of the **Wi-Fi Protected Access (WPA)** protocol, while AES should be part of WPA2.

Note: WiFi and particular vendors uses **different** TKIP/MIC algorithms, which are not compatible. Even WPA was intended to be an intermediate solution because the WiFi only picked a subset of the IEEE 802.11i working draft 3.0).

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)

The Message Integrity Check (MIC) provides data integrity similar to CRC but provides a **non-linear** operation, the "**Michael**" algorithm, and is therefore not vulnerable after RC4 encryption.

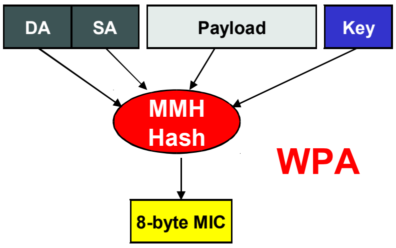
The MIC is based on a **seed** value or a **secret key**, the destination and source MAC, and payload. That is, any change of these values significantly alter the MIC.

The 802.11i task group felt that other commonly used hashing algorithms such as SHA-1 were too computation-intensive to calculate on legacy hardware, so they agreed on the simpler Michael algorithm. Like many hash algorithms, Michael is calculated over the length of the packet, but all of the scrambling it does is based on shift operations and XOR additions, which are quick to calculate. Michael uses a key called the Michael key, which is derived during the WPA procedure (pairwise key).

But according to the 802.11i specification, the Michael algorithm "provides only weak protection against active attack." Therefore **MIC countermeasures** have been specified by the 802.11i: 1) logging and 2) disable and deauthenticate. If two Michael failures occur within one minute, both ends should disable all packet reception and transmission. In addition, the AP should deauthenticate all stations and delete all security associations—a rather drastic solution.

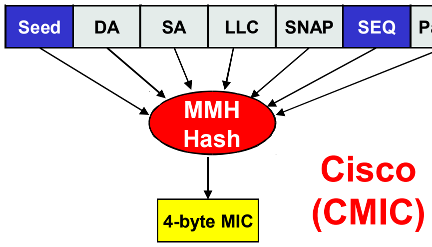
MIC Problems

- 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!!!)



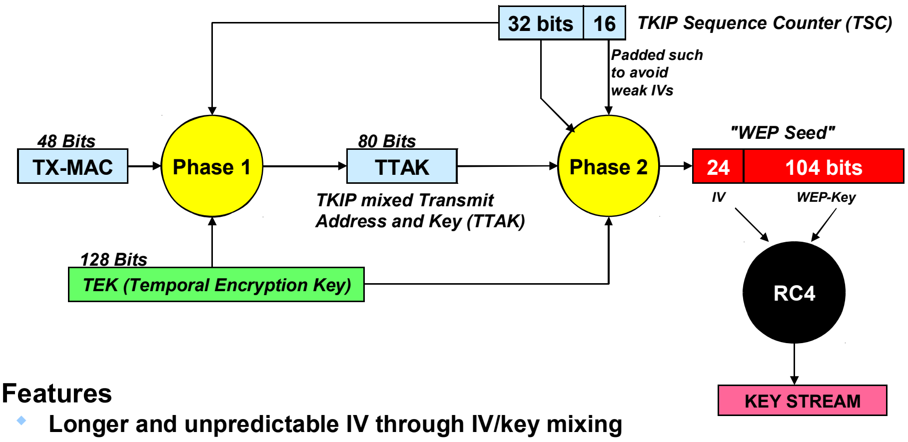
Cisco MIC (CMIC)

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



Note: The Cisco Message Integrity Check serves the same purpose as the 802.11i MIC and is in fact stronger than Michael. It is based on Shai Halevi and Hugo Krawczyk's MMH hashing algorithm.

TKIP (As used by WPA)



- **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**

(C) Herbert Haas 2010/02/15 24

The **WPA's TKIP** solution complies to the 802.11i proposals and uses fresh session keys on every association as well an 48-bit IV space. The mixing functions are based on substitution boxes (S-boxes), which are computationally very efficient, compared to other hash functions.

The **Temporal Encryption Key (TEK)** is derived from the **"Pairwise Master Key"** (**PMK**, also called "base key"), which has been negotiated by the WPA key management protocol. The TEK is used to securely hash a packet counter, the **TKIP Sequence Counter (TSC)**, and the transmit MAC address. A second hash stage enhances the security of the S-box principle.

The TSC is split into 16-bit and 32-bit parts. The 16-bit part is padded to 24 bits to produce a traditional IV. The padding is done in a way that avoids the possibility of weak IV generation. Interestingly, the 32-bit part is not used for the transmitted IV generation; instead, it is utilized in the TKIP per-packet key mixing.

Phase 1 eliminates the use of the same key by all connections, and the second phase reduces the correlation between the IV and per-packet key.

The TSC starts at 0 and increases by 1 for each packet. TSCs must be remembered because they must never repeat for a given key. Each receiver keeps track of the highest value it has received from each MAC address. If it receives a packet that has a TSC value lower than or equal to one it has already received, it assumes it is a rebroadcast and drops it. Thus, packets can only arrive in sequence.

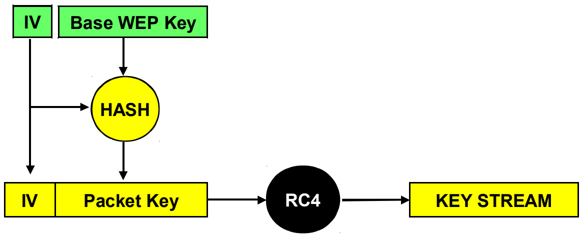
TKIP is only a SW-addon and can reuse the existing WEP hardware.

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.

BTW: TKIP was designed as a 5 year interim solution only! Obviously it will be used much longer than intended.

Cisco TKIP ("CKIP")



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

Because urgent security demands of the market, Cisco developed a proprietary **"Cisco KIP" (CKIP)**, which is based on hashing the static WEP key together with the 24-bit IV to gain the actual packet key.

Also Cisco's solution provides per-packet keys, but **it is recommended to use WPA's TKIP** because:

- WPA's TKIP is computationally more efficient.
- It is more secure, because of the PMK involved.
- The dynamical RC4-key space is much bigger as compared to CKIP.
- Nearly all important vendors support WPA.

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

The estimated weak IV frames appearance interval with TKIP is about a century,
so by the time a cracker collects the necessary 3,000 or more interesting IV frames,
he or she would be 300,000 years old. [Found somewhere: CHECK!]

AES and CCMP

Content

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

This subchapter provides an introduction into 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)

Recently, the **IEEE 802.11i** Security Task Group released two "informative texts" providing WEP hardening: MIC and TKIP. The IEEE 802.11 Task Group "i" is working on standardizing WLAN encryption improvements. Two new network types, called Transition Security Network (TSN) and Robust Security Network (RSN) had been defined.

The Temporal Key Integrity Protocol (TKIP, initially referred to as WEP2) is an interim solution (as part of TSN) that fixes the key reuse problem of WEP.

TKIP is a compromise on strong security and possibility to use existing hardware. Still uses RC4 but per-packet keys plus replay protection through a keyed packet authentication mechanism (Michael MIC).

TKIP begins with a 128 bit "temporal key" shared among clients and access points. TKIP combines the temporal key with the client's MAC address and then adds a 6-byte IV to produce the key that will encrypt the data. Thus each station uses different key streams for encryption. TKIP changes keys every 10,000 packets, using a dynamic distribution method.

The IEEE specifies to use the **Advanced Encryption Standard (AES)** instead of RC4 for TKIP in the long run (RSN), combined with Counter Mode - Cipher Block Chaining - Message Authentication Code (CBC MAC) to provide strong integrity and message authentication. Also the term "Wireless Robust Authenticated Protocol" (WRAP) is sometimes used synonymously for this concept.

The **Wi-Fi** specified TKIP and MIC as mandatory features of the **Wi-Fi Protected Access (WPA)** protocol, while AES should be part of WPA2.

Note: WiFi and particular vendors uses **different** TKIP/MIC algorithms, which are not compatible. Even WPA was intended to be an intermediate solution because the WiFi only picked a subset of the IEEE 802.11i working draft 3.0).

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 !!!

How secure is AES compared to RC4?

RC4 uses up to 128 bits key length, AES uses 256 bits, that is the AES key is 128 bits longer. If only brute force attacks are assumed (algorithms are save enough) and considering Moore's law (computing power doubles every 18 month), then AES is at least $\log_2(128)*18$ months ahead, that is more than 10 years, compared to RC4.

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

The **802.11i** standard was finished in May 2004 and **approved in June 2004**. The main result, WPA2, includes support for more robust encryption algorithm (CCMP: AES in Counter mode with CBC-MAC) to replace TKIP and **optimizations for handoff** (reduced number of messages in initial key handshake, pre-authentication, and PMKSA caching).

The **Advanced Encryption Standard (AES)** is considered as state-of-the-art encryption method, designed recently, using Rijndael as algorithm and is official successor of DES or 3DES. This 128-bit block cipher is considered unbreakable for the next ten years or so.

CCM is a actually the block cipher *mode* of AES that provides both encryption and authentication. It is a combination of counter-mode encryption and CBC-MAC authentication which are two modes that have been studied extensively for many years. CCM was developed as a non-patented alternative to OCB ("Offset Codebook") for use in secure wireless networks, but it can be used in almost any situation that requires secure communications. With CCM encryption and authentication

Links:

Rijndael description and algorithm:
http://csrc.nist.gov/CryptoToolkit/aes/rijndael/

AES Lounge:
http://www.iaik.tu-graz.ac.at/research/krypto/AES/

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 $1 \leq j \leq t$, $x_j \leftarrow c_{j-1} \oplus E_K^{-1}(c_j)$.

Although CBC mode decryption recovers from errors in ciphertext blocks, modifications to a plaintext block xj during encryption alter all subsequent ciphertext blocks. This impacts the usability of chaining modes for applications requiring random read/write access to encrypted data.

An exposed IV might allow a man-in-the-middle (MITM) to change the IV value in-transit. Changing the IV changes only the deciphered plaintext for the first block, without garbling the second block. Any or all bits of the first block plaintext can be changed systematically with complete control.

The most obvious way to prevent deliberate MITM changes to the first block plaintext with the IV is to encipher the IV; that prevents an opponent from changing plaintext bits systematically.

Counter Mode (CCM)

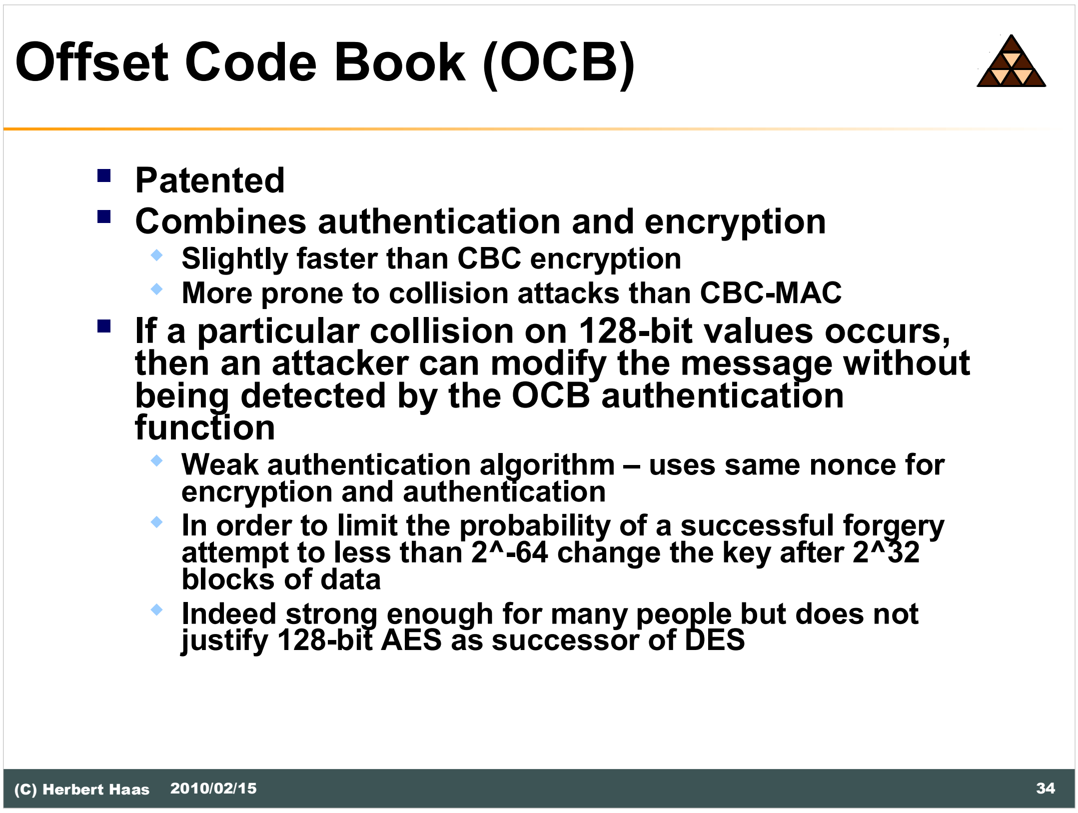
- **Instead of directly encrypting the data only a counter is encrypted**
- **Message is then XORed with this encrypted counter**
- **Counter = nonce (SQNR, Source-MAC, Priority fields)**

WPA2 supports **FIPS 140-2** compliant security, basically AES in counter mode. (An early draft included AES-OCB instead but it was dropped due to patent issues.) A 48 bit IV protects against replay attacks.

Authentication and Integrity is maintained using an **8 byte CBC-MAC** with a 48 bit nonce. Besides the data also the source and destination MAC addresses in the header are protected by the CBC-MAC. (These fields are called Additional Authentication Data (AAD).

The CBC-MAC, the nonce, and additional 2 byte IEEE 802.11 overhead make the CCMP packet 16 octets larger than an unencrypted IEEE 802.11 packet.

The AP advertises cipher suites both in beacons and probe responses.



AES-OCB is a mode that operates by augmenting the normal encryption process by incorporating an offset value.

The routine is initiated with a unique nonce (the nonce is a 128-bit number) used to generate an initial offset value. The nonce has the XOR function performed with a 128-bit string (referred to as value L).

The output of the XOR is AES-encrypted with the AES key, and the result is the offset value.

The plain-text data has the XOR function performed with the offset and is then AES-encrypted with the same AES key.

The output then has the XOR function performed with the offset once again. The result is the cipher-text block to be transmitted.

The offset value changes after processing each block by having the XOR function performed on the offset with a new value of L.

See http://www.cs.ucdavis.edu/~rogaway/ocb/index.html

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:

$$S := M_1 \oplus \cdots \oplus M_{m-1} \oplus C_m 0^* \oplus Y_m$$ **… "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)**

$C_m 0^*$ … last ciphertext block padded with zeros to full 128 bit length

802.11 Standard Authentication

(C) Herbert Haas 2010/02/15

Content

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

This subchapter provides an introduction into the 802.11 standard authentication methods.

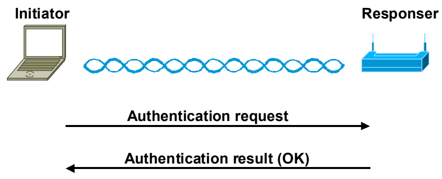
Objective

After completing this chapter the following tasks could be solved:

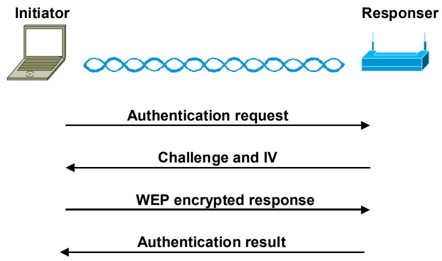
- Highlight the design flaws of the WLAN standard authentication
- Explain the design idea of 802.1x
- Compare EAP-TLS, LEAP, PEAP, EAP-TTLS, EAP-FAST with each other and emphasize important security features
- Explain the design concept of WPA and WPA2
- Implement a reliable 802.1x infrastructure over a WAN connection
- List important issues to be considered when choosing a VPN design
- Explain PSPF

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…



Open System Authentication allows anyone to gain access to the WLAN. It is generally applicable where public access should be provided, for example in universities, airports, or hotels. The authentication process is realized using "management" frames with "authentication" as subtype. Specifically, the open system method is indicated using an algorithm identification field.

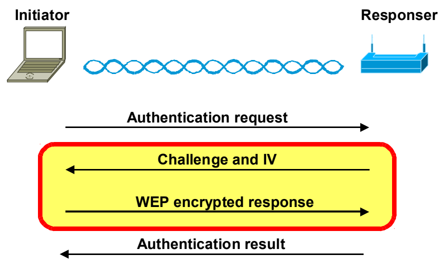
Shared Key Authentication uses the WEP algorithm to implement a four-step handshake procedure, provided that each user has the same shared key. Shared Key Authentication only enables client authentication but the client can never be sure whether the AP is a "rouge" AP. Furthermore, WEP is vulnerable, and hence this authentication process can be attacked.

This four-step procedure requires WEP support from both sides. It is assumed that both sides possess the same shared key. The initiator sends an authentication request management frame indicating that it wish to use “shared key” authentication. The responder replies by sending an authentication management frame containing an 128 octets challenge text. This challenge text is generated by using the WEP pseudo-random number generator (PRNG) with the “shared secret” and a random IV. The initiator receives the challenge and the IV and sends a WEP-encrypted version of the challenge back to the responder, hereby using the shared secret and the IV. The responder decrypts the received frame and verifies the 32-bit CRC integrity check and that the challenge text matches that sent in the first message. In this case the authentication is successful and the responder completes the process by sending the authentication result. Optionally, the initiator and the responder switch roles and repeat the process to ensure mutual authentication. However, mutual authentication is seldom implemented. The value of the status code field is set to zero when successful, and to an error value if unsuccessful. The element identifier identifies that the challenge text is included. The length field identifies the length of the challenge text and is fixed at 128. The challenge text includes the random challenge string.

Besides WEP design flaws, the whole authentication is tied to the device identity, not the user's identity. That is, a stolen device can be abused to gain access to the WLAN.

Shared Key Authentication

- Attacker captures 2nd and 3rd authentication message and has
  - Plaintext P (the challenge)
  - Ciphertext C = $RC4^K(P)$
- The keystream is simply $S = C \oplus P$
- 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 !!!



(C) Herbert Haas 2010/02/15 38

Never use Shared Key Authentication

An attacker could easily capture the 2nd and 3rd authentication messages and possesses a plaintext (the challenge) and the corresponding ciphertext. Remember that the keystream S can be easily calculated by XORing both messages.

Other fields (besides the challenge) are rather static and can be guessed—they have always the same values in each authentication process.

Having S, an attacker can easily authenticate to the network as he is able to correctly respond to each challenge sent by a responder.

802.1x and EAP Authentication

Content

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

This subchapter provides an overview of 802.1x authentication and various EAP protocols.

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)*

The IEEE is working on a supplement to the 802.1d standard which will define the changes necessary to the operation of a MAC layer bridge in order to provide **port-based network access** control capability. This standard is known as 802.1x and has been adopted by the 802.11i working group.

802.1x provides port-based access control, that is, a special authentication mechanism is used to switch a bridge port or the AP from an unauthorized state into an authorized state. Only the latter state allows traffic other than 802.1x traffic.

Using 802.1x, a wireless client that associates with an AP cannot gain access to the network until the user performs a network logon or provides other strong credentials. Practically, when the user enters a username and password into a network logon dialog box or its equivalent, the client and an authentication server, a RADIUS server, perform a **mutual authentication**. Additionally, the RADIUS-based authentication server (AS) allows **centralized user credential management**.

Note that the AP acts as pass-through device, while the actual authentication process is performed by the authentication server. The authentication server and client then derive a client-specific WEP/TKIP key to be used by the client for the current logon session. User passwords and session keys are never transmitted in the clear, over the wireless link.

The whole authentication process is conducted by the **Extensible Authentication Protocol (EAP)** which has been defined in RFC 2284 as PPP extension. Note that EAP is only a meta-authentication protocol. EAP initiates the process and carries the actual authentication protocol, for example the Transport Layer Security (TLS) protocol and others. Most of them provide a session identifier and therefore provide seamless handover between access points, without re-authentication need.

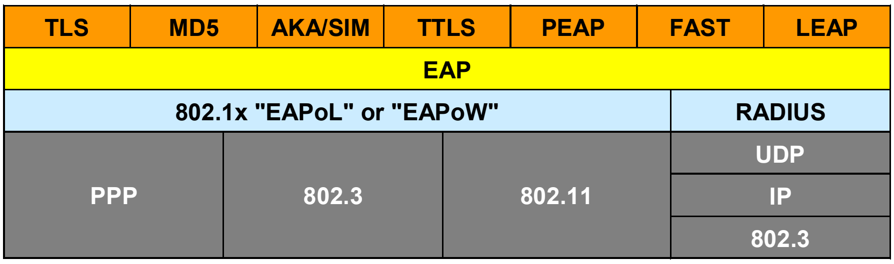
Note that 802.1x can only negotiate per-user session keys for unicast transmission. A single **static broadcast key** must also be configured on an access point for 802.1x clients to receive broadcast and multicast messages. This is typically performed automatically.

Reauthentication can be easily realized, because each AP can ask the central AS whether the client is already authenticated. This principle supports fast roaming (even better, if there is a caching instance in-between).

Wi-Fi's WPA also requires 802.1x as authentication method.

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



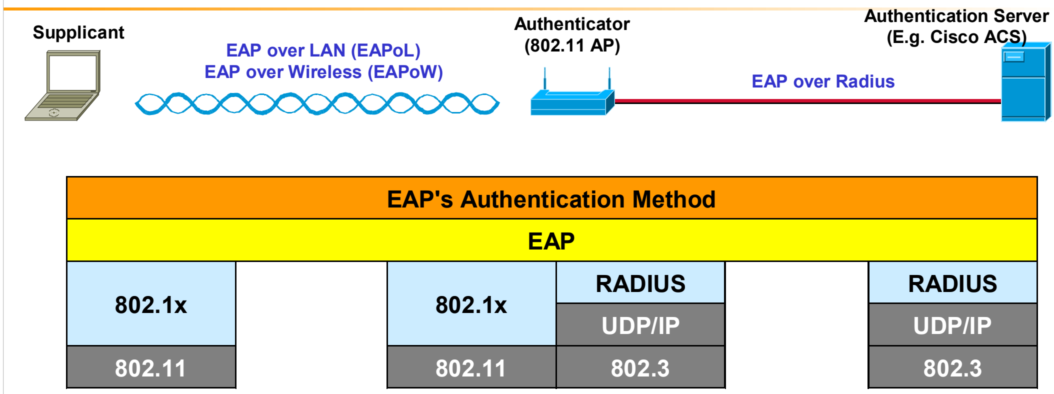
802.1x relies on EAP as underlying authentication protocol carrier. EAP is **extensible**, as it allows to develop and deploy new authentication protocols easily without changing the AP software. That is, EAP can be imagined as a container for authentication schemes.

The picture above shows the layers involved. EAP itself is either carried by a layer-2 protocol such as 802.3 ("EAP over LAN", **EAPoL**) or 802.11 ("EAP over Wireless", **EAPoW**), or by RADIUS ("**EAP over RADIUS**").

In order to be carried over RADIUS, the EAP information is decomposed into information elements and additionally, new Attribute Value Pairs (AVPs) had to be defined ("eap-radius").

See RFC 2284 for further details.

802.1x – Protocol Layers



- 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

Each 802.1x-based authentication consists of **three participants**:

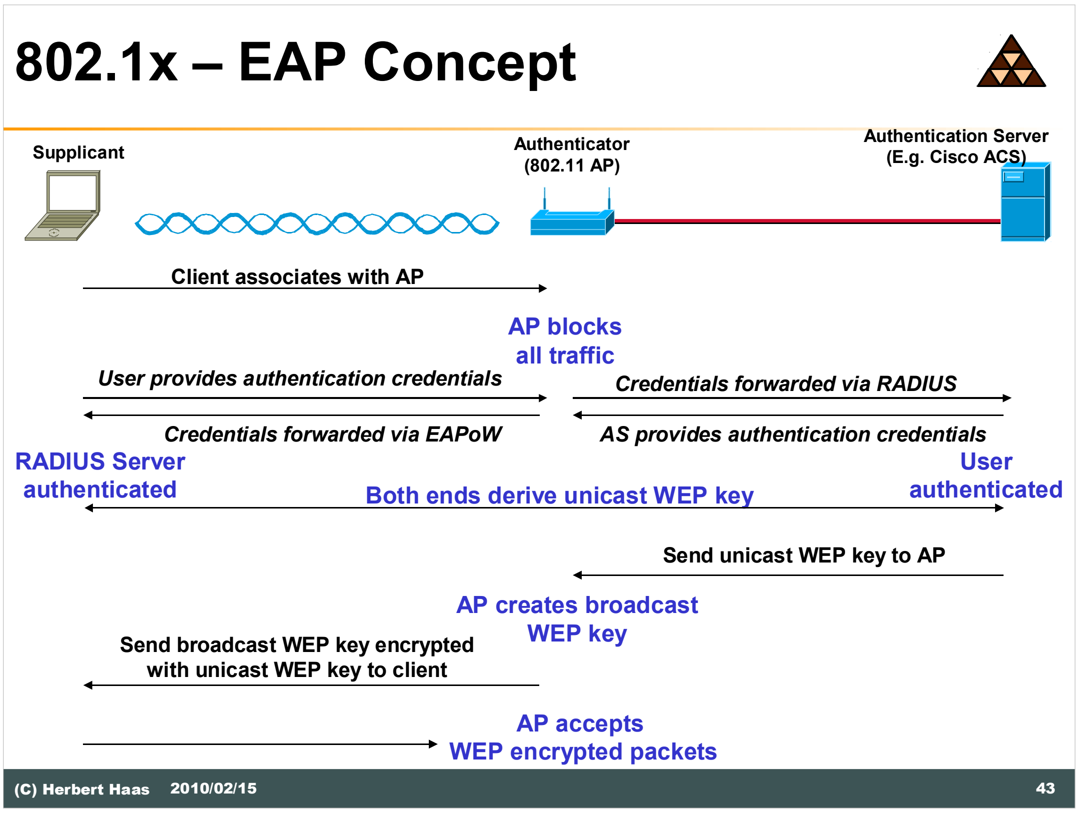
1. The client, who is called "**Supplicant**"
2. The "**Authenticator**", which is actually the AP
3. An "**Authentication Server**" which must support eap-radius.

Both the Supplicant and the Authentication Server are authenticated to each other but this handshake is intercepted by the Authenticator, which forwards these messages to the endpoints.

Of course also the Authenticator must be authenticated. This is typically done by a **shared secret** between the Authenticator and the Authentication Server.

Note: The Authenticator is basically an 802.1x-to-UDP "bridge".

When an 802.1X-capable host starts up, it will initiate the authentication phase by sending the EAPoL-Start 802.1x protocol data unit (PDU) to the reserved IEEE multicast MAC address (01-80-C2-00-00-03) with the Ethernet type or length set to **0x888E**.



This picture illustrates the basic concept of 802.1x and EAP.

1) When the AP receives an association request from the client, the AP requires the client to authenticate via EAP.

2) The client sends his/her authentication credentials, the AP cannot verify these by itself but forwards them to a preconfigured authentication server (a RADIUS server).

3) The RADIUS server finds this user in its database and verifies the correctness of the associated credentials.

4) During this EAP negotiation, also the client can authenticate the RADIUS server—and by doing this, also the AP is authenticated implicitly (because the AP and the RADIUS server are bound via a shared secret.

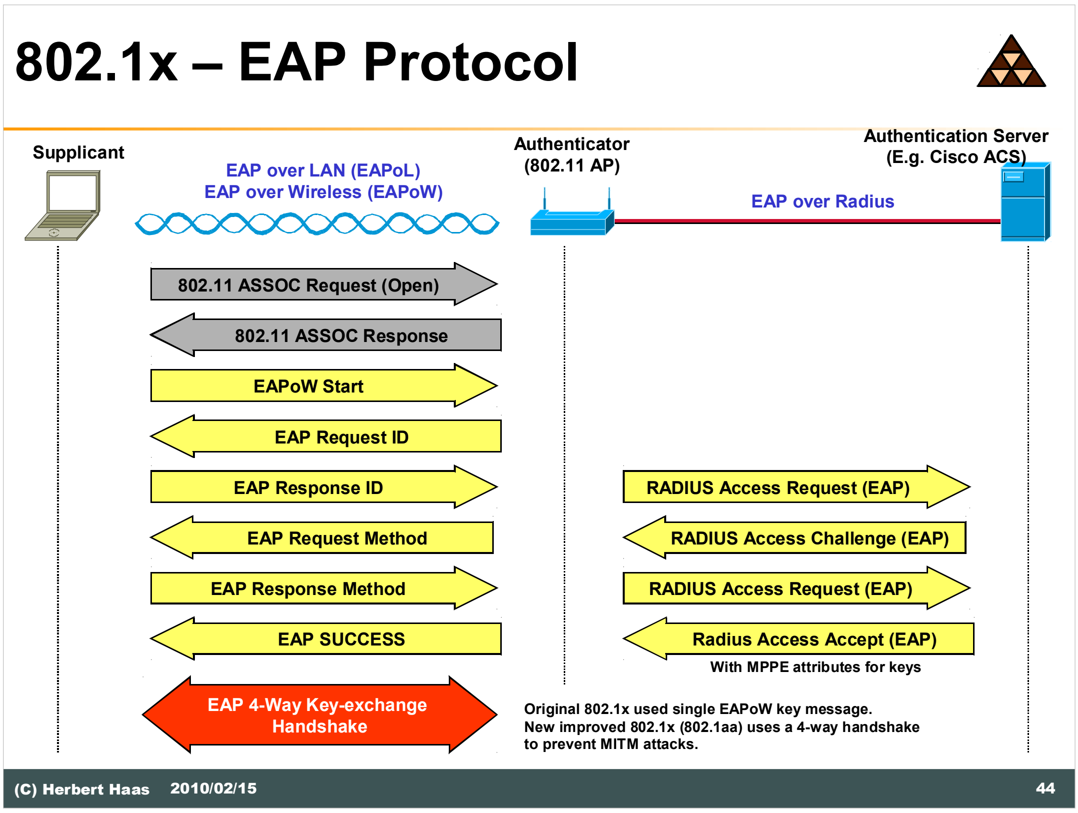
5) Both client and RADIUS server determine a unicast WEP key.

6) The AP sends this unicast WEP key to the AP.

7) The AP creates a random broadcast WEP key, encrypts it using the unicast WEP key and forwards it to the client.

8) Now, the client and the AP can communicate using WEP encrypted packets. The AP will decrypt each correctly encrypted packet from the client and will forward it to the wired LAN.

Note: Cisco supports **broadcast key rotation**. After a certain amount of time the AP dynamically distributes a new broadcast key to the clients. Obviously this feature is only possible when EAP is enabled.



EAP provides an envelope that can carry many **different kinds of authentication types**: challenge/response, one time passwords (OTPs) , SecurID tokens, digital certificates, etc. What exactly happens between "EAP Start" and "EAP Success" depends upon the type of authentication being used.

The original 802.1X standard used a single EAPoW Key message for this purpose, but the **new improved 802.1x** (called 802.1aa) uses a **four-way handshake to prevent man-in-the-middle attacks** that might otherwise compromise these keys. After both ends —the client and the AP—of the wireless association have session keys, data sent over the air can be encrypted to prevent eavesdropping.

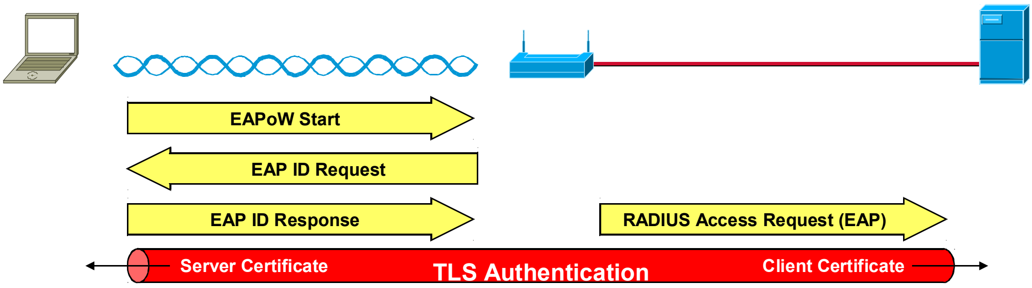
One of the most important benefits will be felt by users who are **roaming** within an organization's wireless LAN and require a seamless connection. If they are asked to authenticate themselves each time they pass from one conference room to another, they will want to give up security in favor of convenience.

Using the connection re-establishment mechanism provided by the TLS handshake users can have one seamless connection while roaming between different APs connected to the same backend server. If the **session ID** is still valid, the wireless client and server can share previously negotiated secrets to establish a new handshake and keep the connection alive.

Additionally, secure **session timeouts** trigger re-authentication and new WEP (TKIP) keys.

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



The **TLS Working Group** was established in 1996 to standardize a "transport layer" security protocol. The working group began with SSL version 3.0, and in 1999, RFC 2246, the "TLS Protocol Version 1.0" was published as a Proposed Standard. The working group has also published RFC 2712, "Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)" as a Proposed Standard, and two RFCs on the use of TLS with HTTP.

EAP-TLS was the first **most-widely implemented** 802.1x method for WLANs. EAP-TLS supports **session expiration** and 802.1x re-authentication by using the RADIUS session timeout option (RADIUS Internet Engineering Task Force option 27). To avoid IV reuse (IV collisions), the base **WEP key is rotated** before the IV space is exhausted.

However, **several problems lead to a seldom use** of EAP-TLS:

• Every client needs a certificate. This is only affordable if a PKI is available providing a CA for certificate management, revocation and so on.

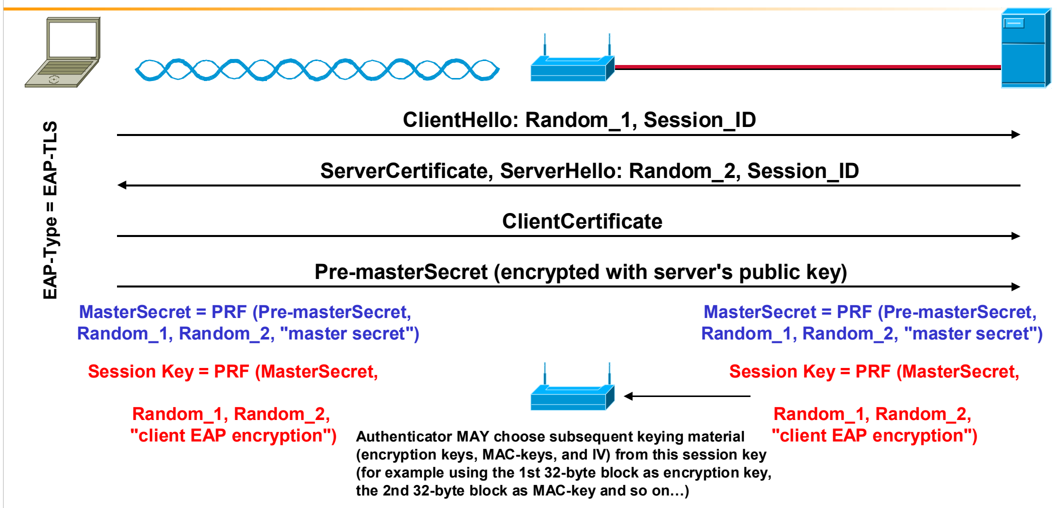
• In the three messages of the EAP starting sequence, the user-ID is revealed. This is considered privacy-critical today.

• Fast session reconnection (for VoIP) is not possible.

Remember: A certificate is a cryptographically signed structure, that guarantees the association between at least one *identifier* and a *public key*.

Note: The name in the client certificate must be the same username as in the AS user database. This is one important reason to choose a private Root-CA, besides the advantage of more control.

802.1x – EAP-TLS (2)



- 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, ...)

The Session_ID can be used for fast re-authentication purposes.

As part of the TLS handshake between the server and the client, the client generates a **pre-master secret** and encrypts it with the server's public key. Then this pre-master secret is sent to the AS. Another option would be to use Diffie-Hellman exchange to derive the pre-master secret.

The pre-master secret, server and client random values, and "master secret" string value are used to generate a **master secret** per session. A Pseudo Random Function (PRF) is used again along with master secret, client and server random values, and "client EAP encryption" string value to generate the 128-bit **session keys**, Message Authentication Code (MAC) keys and initialization values (for block ciphers only).

Note that both the client and the AS independently derive the session keys. However, the length of the session key is determined by the authenticator (the AP) and is sent in the EAPoL key message at the end of the EAP authentication to the client.

A TLS session is governed by a security context, which consists of session identifier, peer certificate, compression method, cipher spec for the session key, MAC algorithm parameters, and the shared master secret.

TLS sessions expire after some time and the AS can be notified via RADIUS.

EAP-TLS is natively supported in MAC OS 10.3 and above, Windows 2000 SP4, Windows XP, Windows Mobile 2003 and above, and Windows CE 4.2

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)**

Cisco's **Lightweight EAP (LEAP)** implementation is widely deployed because of its simplicity as it is based on **shared user secrets**. Furthermore, only LEAP supports **fast secure roaming**, necessary if low-delay applications are used (e. g. VoIP).

Cisco has developed drivers for most versions of Microsoft Windows (Windows 95, 98, Me, 2000, NT and XP) and uses the **Windows logon** as the Cisco LEAP logon.

A software shim in the Windows logon allows the username and password information to be passed to the Cisco Aironet client driver. The driver will convert the password into a Windows NT key and hand the username and Windows NT key to the Cisco NIC. The NIC executes 802.1x transactions with the AP and the authentication, authorization, and accounting (AAA) server.

Note: Neither the password nor the password hash is ever sent across the wireless medium.

Additionally, any Open Database Connectivity (ODBC) that uses MS-CHAP passwords can also be used with LEAP.

Note: If an AS is used for both Cisco LEAP and MAC authentication, the MAC address should use a different strong password for the required MS-CHAP/CHAP field. If not, an eavesdropper can spoof a valid MAC address and use it as a username and password combination for Cisco LEAP authentication.

Note: The LEAP key generation mechanism is proprietary and is generated every (re)authentication, thus achieving key rotation. The session timeout in RADIUS allows for periodic key rotation, thus achieving security against sniffing and hacking the keys. The RADIUS exchanges for LEAP include a couple of Cisco-specific attributes in the RADIUS messages.

To avoid IV reuse (IV collisions), LEAP rotates the base WEP key before the IV space is exhausted.

Note: LEAP is only as strong as the passwords used. Therefore it is vulnerable to dictionary attacks. At least 10-character passwords should be used.

BTW: Implementation details of LEAPv1: ChallengeLEN=8, RESPONSE_LEN=24, KEY_LEN=16 [BYTES]

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

The 8 Byte challenge is encrypted 3 times, using Seed3 for the third DES encryption. Since the attacker knows the challenge and the encrypted response, a simple brute force attack quickly recovers seed3. Now the search duration in the attacker's dictionary file can be significantly reduced. Assuming that this dictionary file has been prepared such that it already contains the NT hashes of each password, the lookup algorithm must only look for hashes for which bytes 15 and 16 matches the recovered seed3.

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

Example: Asleap, cracking password "test"

(C) Herbert Haas 2010/02/15 49

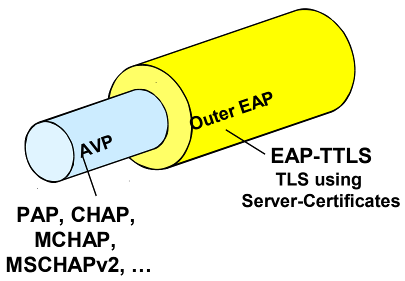
A good policy should require a password length of at least 12 characters, including numbers, mixed case, and punctuation. It should also include a requirement that passwords be based on neither words found in any dictionary nor any variant of the username.

There are cracking dictionaries for hundreds of languages and commonly used words, such as names of places, people, and movies. Usually the only way to enforce strong passwords is with tools that enforce passwords at creation time. Users are good at choosing easy-to-remember passwords and tend to ignore unenforced rules. It is a good idea to run regular, automated password cracking on your organization's passwords and warn users or disable accounts with bad passwords. Your organizational environment determines what strength of password enforcement and frequency of password changes is acceptable to your user community.

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:



EAP-TTLS was developed by Funk Software and Certicom, and was first supported by Agere Systems, Proxim, and Avaya. Today EAP-TTLS is being considered by the IETF as a new standard.

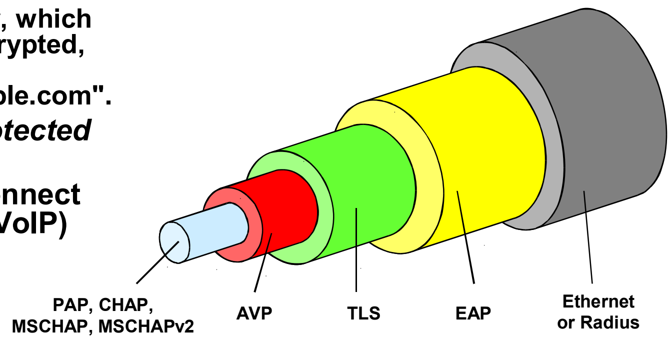
The structure of **Tunnelled TLS (TTLS)** and PEAP are **quite similar**. Both are two-stage protocols that establish security in stage one and then exchange authentication in stage two.

Stage one of both protocols establishes a TLS tunnel and authenticates the authentication server to the client with a **certificate**. Once that secure channel has been established, client authentication credentials are exchanged in the second stage.

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)**

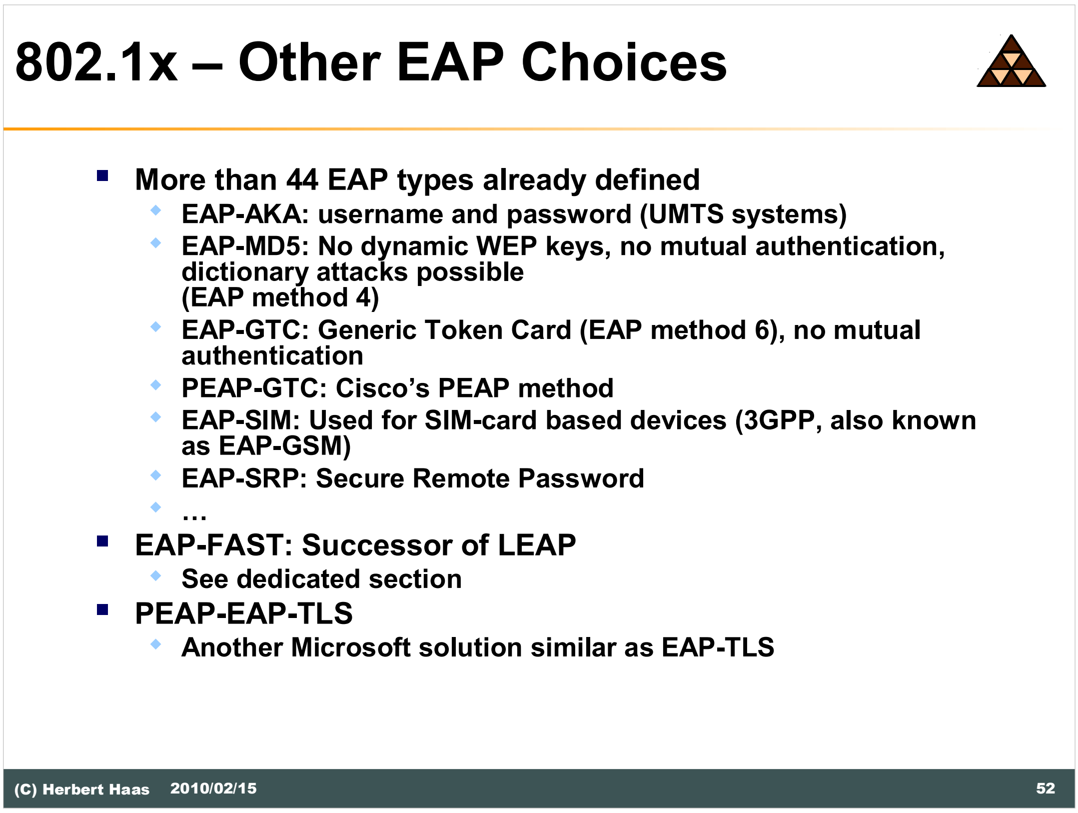
Detailed:



Other than PEAP, EAP-TTLS supports **any authentication method**, not only EAP-methods. Therefore, there is no inner EAP session but **RADIUS-like AVPs** are used to carry the authentication data.

EAP-TTLS often uses PAP (also with Linux).

As with PEAP, **user identity information is protected.**



There are other EAP methods which are currently not so important in the 802.11 WLAN world.

EAP-AKA works similar as LEAP. AKA stands for Authentication and Key Agreement. It is also used with HTTP Authentication and GSM. See *draft-arkko-pppext-eap-aka-12.txt* for details.

EAP-MD5 does not support mutual authentication and is not strong enough, also some vendors use it with WLAN devices.

EAP-GTC is typically only used as inner EAP-method of PEAP. In this case it is often called "PEAP-GTC".

EAP-SIM is used by 3GPP applications (GSM and UMTS). SIM stands for Subscriber Identity Module.

EAP-SRP (Secure Remote Password) is a method used by some vendors, mainly Orinoco.

WPA-Note: EAP-MD5, EAP-GTC, EAP-OTP, and EAP-MSCHAPV2 cannot be used alone with WPA. They can only be used as inner authentication algorithms with EAP-PEAP and EAP-TTLS.

Microsoft supports another form of PEAPv0 (which Microsoft calls **PEAP-EAP-TLS**) that Cisco and other third-party server and client software don't support.

PEAP-EAP-TLS does require a client-side digital certificate located on the client's hard drive or a more secure smartcard. PEAP-EAP-TLS is very similar in operation to the original EAP-TLS but provides slightly more protection due to the fact that portions of the client certificate that are unencrypted in EAP-TLS are encrypted in PEAP-EAP-TLS.

Since few third-party clients and servers support PEAP-EAP-TLS, users should probably avoid it unless they only intend to use Microsoft desktop clients and servers.

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

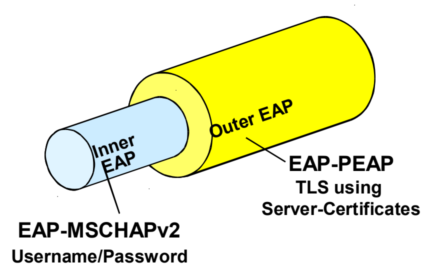
This list is just for reference.

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:



Protected EAP (PEAP) has been developed by **Cisco and Microsoft** and is only available on the newest Microsoft platforms (XP).

PEAP is a **two-stage protocol** that establish a secure TLS tunnel which carries an **inner EAP session**.

PEAP only supports EAP-type authentication. Microsoft proposes MS-CHAPv2, while Cisco prefers Generic Token Cards (EAP-GTC). Cisco differentiates "v0" and "v1" while Microsoft only knows "PEAP", which means PEAPv0 and only supports MSCHAPv2. Cisco's implementation "v0" also supports EAP-SIM, while "v1" also supports EAP-GTC.

The main advantage of PEAP is that **client certificates are not necessary**.

Support for MS-DB but no support for LDAP-DB.

The PEAP result is the so-called **Compound Session Key (CSK)** which is actually a concatenation of the Master Session Key (MSK), which is 64 bytes, and the Extended Master Session Key (EMSK), which is 64 bytes.

The MSK and EMSK are defined in RFC 3269 (also known as RFC 2284bis) as follows:

> MSK: Key derived between the peer and the EAP server and exported to the authenticator.
>
> EMSK: Additional keying material derived between the peer and the EAP server and exported to the authenticator. It is reserved for future use and not defined in the current RFC. In addition, the PEAP key mechanisms are designed for future extensibility; the exchange sequences (and choreographies) and formats can be used for handling any key material; binding inner, outer, and other intermediate methods; and verifying the security between the layers that are required for future algorithms.

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"**

 56

The PEAP result is the so-called **Compound Session Key (CSK)** which is actually a concatenation of the Master Session Key (MSK), which is 64 bytes, and the Extended Master Session Key (EMSK), which is 64 bytes.

The MSK and EMSK are defined in RFC 3269 (also known as RFC 2284bis) as follows:

> MSK: Key derived between the peer and the EAP server and exported to the authenticator.
>
> EMSK: Additional keying material derived between the peer and the EAP server and exported to the authenticator. It is reserved for future use and not defined in the current RFC. In addition, the PEAP key mechanisms are designed for future extensibility; the exchange sequences (and choreographies) and formats can be used for handling any key material; binding inner, outer, and other intermediate methods; and verifying the security between the layers that are required for future algorithms.

Note:

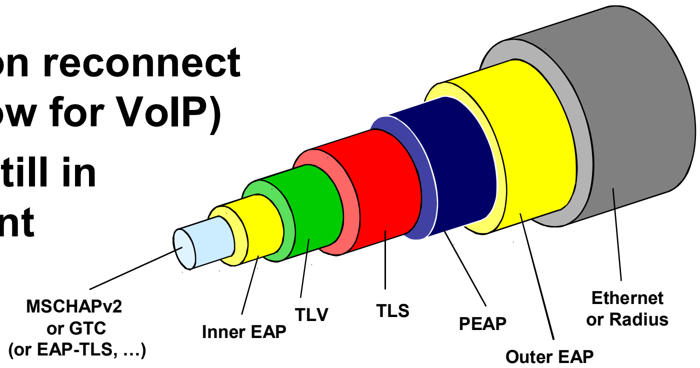
PEAPv0 and PEAPv1 both refer to the outer authentication method and is the mechanism that creates the secure TLS tunnel to protect subsequent authentication transactions while EAP-MSCHAPv2, EAP-GTC, and EAP-SIM refer to the inner authentication method which facilitates user or device authentication. PEAPv0 supports inner EAP methods EAP-MSCHAPv2 and EAP-SIM while PEAPv1 supports inner EAP methods EAP-GTC and EAP-SIM. Since Microsoft only supports PEAPv0 and doesn't support PEAPv1, Microsoft simply calls PEAPv0 PEAP without the v0 or v1 designator. Another difference between Microsoft and Cisco is that Microsoft only supports PEAPv0/EAP-MSCHAPv2 mode but not PEAPv0/EAP-SIM mode.

However, Microsoft supports another form of PEAPv0 called **PEAP-EAP-TLS** that Cisco and other third-party server and client software don't support. PEAP-EAP-TLS does require a client-side digital certificate located on the client's hard drive or a more secure smartcard. PEAP-EAP-TLS is very similar in operation to the original EAP-TLS but provides slightly more protection due to the fact that portions of the client certificate that are unencrypted in EAP-TLS are encrypted in PEAP-EAP-TLS. Since few third-party clients and servers support PEAP-EAP-TLS, users should probably avoid it unless they only intend to use Microsoft desktop clients and servers.

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



Security Claims of PEAPv2

Intended use: Wireless or Wired networks, and over the Internet, where physical security cannot be assumed.

Auth. mechanism: Use arbitrary EAP and TLS authentication mechanisms for authentication of the client and server.

Ciphersuite negotiation: Yes.

Mutual authentication: Yes. Depends on the type of EAP method used within the tunnel and the type of authentication used within TLS.

Integrity protection: Yes

Replay protection: Yes

Confidentiality: Yes

Key derivation: Yes

Key strength: Variable

Dictionary attack prot: Not susceptible.

Fast reconnect: Yes

Crypt. binding: Yes.

Acknowledged S/F: Yes

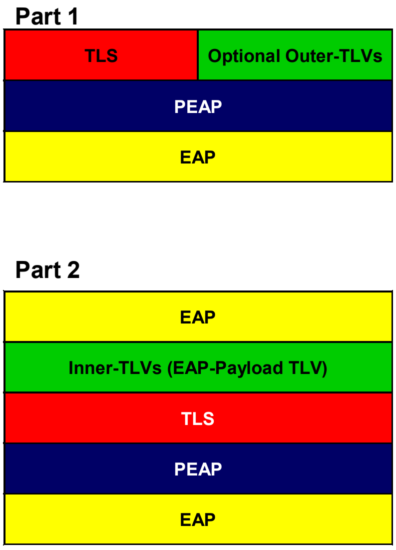
Session independence: Yes.

Fragmentation: Yes

State Synchronization: Yes [80211Req]

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



The TLS v1.0 mandatory-to-implement ciphersuite TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA must be supported

For light-weight devices also other TLS cipher suites supported

PEAPv2 client and servers SHOULD support

TLS_RSA_WITH_3DES_EDE_CBC_SHA

TLS_RSA_WITH_RC4_128_MD5

TLS_RSA_WITH_RC4_128_SHA

TLS_RSA_WITH_AES_128_CBC_SHA

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

Unfortunately many people use PEAP inside a "server-unauthenticated TLS tunnel" which is (unfortunately) a supported method – but this actually conflicts with the initial idea of a secure-tunnel authentication!

Therefore always install appropriate root certificates!

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)

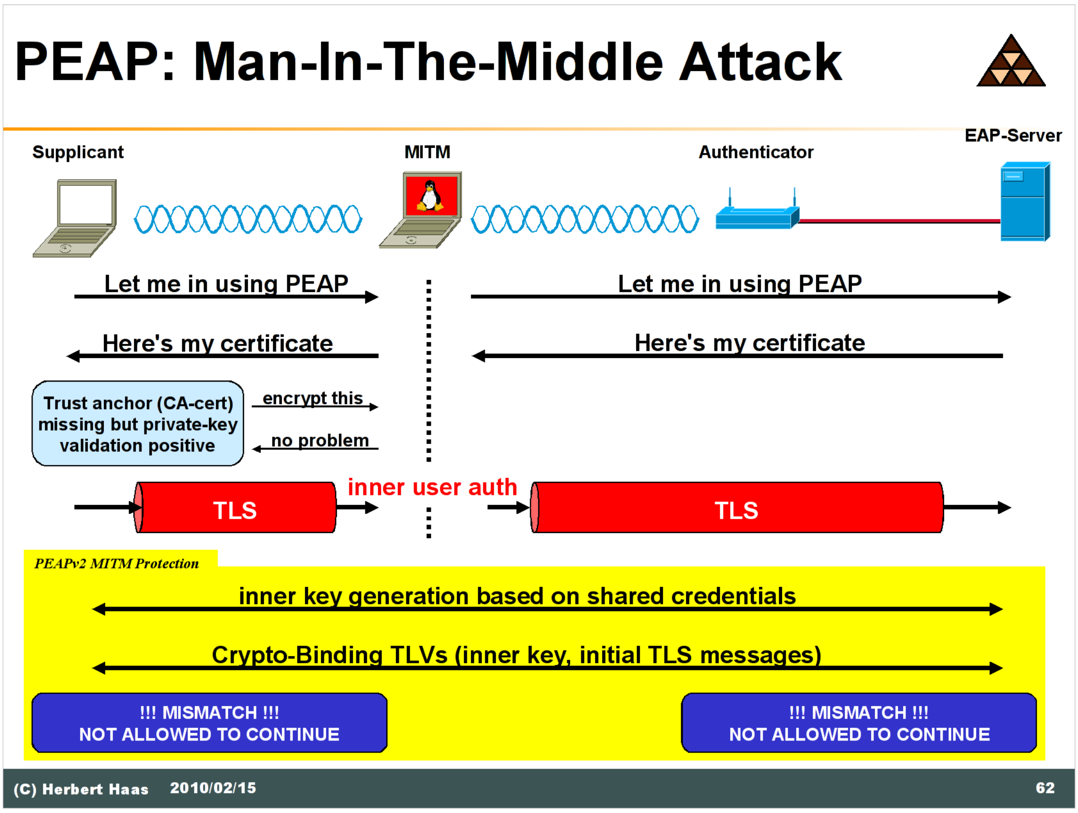
If the validation of the server certificate fails (because of failing private key validation or invalid certificate parameters) then the "provisioning inside a server-unauthenticated TLS tunnel"-mode must not be entered.

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 !!!

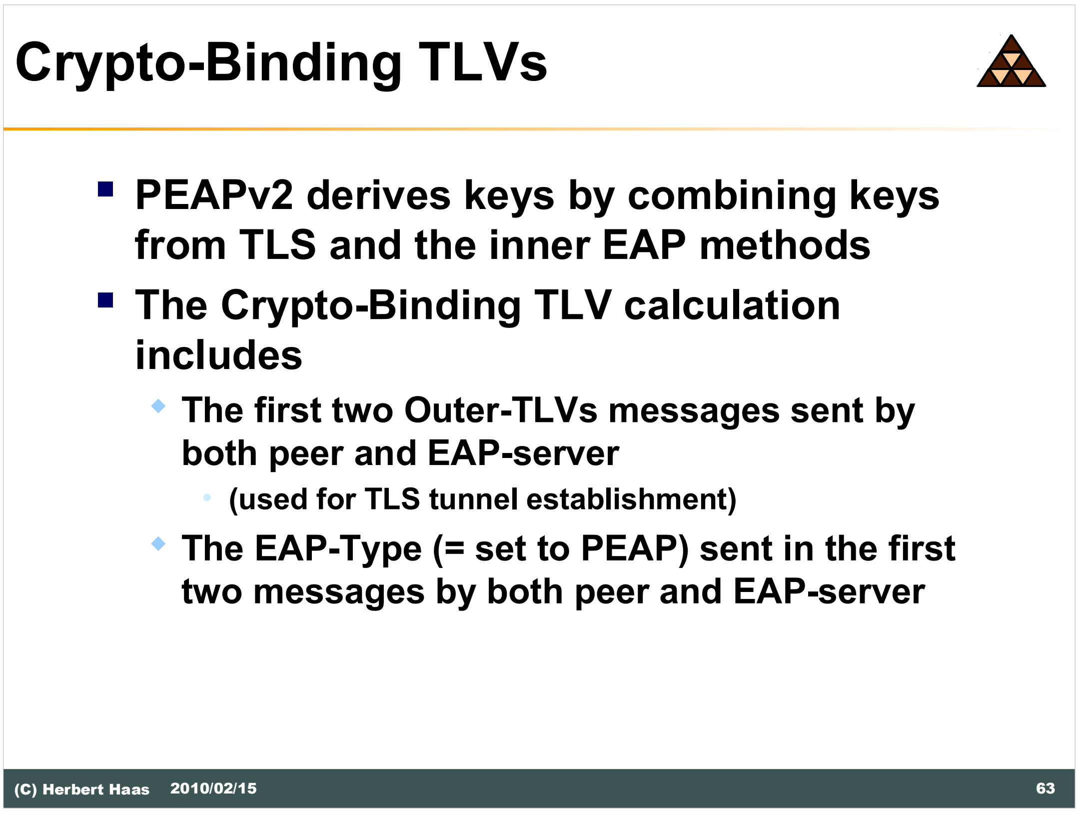
Note that with every EAP method, there must be a final EAP success/failure indication sent in clear – to inform the authenticator.

An early Cisco solution to the MITM problem with pre-PEAPv2 versions is to enforce the client to choose a PEAP trust anchor. That is the client must select a root certificate issuer from a list. If the certificate offered by the server cannot be validated via the pre-selected trust anchor, the authentication process stops. Unfortunately, also "any" can be selected.



A man-in-the-middle can spoof the client to authenticate to it instead of the real EAP server; and forward the authentication to the real server over a protected tunnel. Since the attacker has access to the keys derived from the tunnel, it can gain access to the network.

PEAP version 2 prevents this attack by using the keys generated by the inner EAP method in the crypto-binding exchange described in protected termination section. This attack is not prevented if the inner EAP method does not generate keys or if the keys generated by the inner EAP method can be compromised. Hence, in cases where the inner EAP method does not generate keys, the recommended solution is to always deploy authentication methods protected by PEAPv2.



Outer-TLVs SHOULD NOT be included in other PEAP packets since there is no mechanism to detect modification.

For *subsequent* packets (after the first two) the EAP Type in the clear could be modified and will likely result in failure, hence it is not included in the Crypto-Binding calculation.

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 "fast reconnect" is desirable in applications such as wireless roaming, since it minimizes interruptions in connectivity. It is also desirable when the "inner" EAP mechanism used is such that it requires user interaction. The user should not be required to re-authenticate herself, using biometrics, token cards or similar, every time the radio connectivity is handed over between access points in wireless environments.

Since PEAPv2 Part 1 may not provide client authentication, establishment of a TLS session (and an entry in the TLS session cache) does not by itself provide an indication of the peer's authenticity. Implementations that do not remove TLS session cache entries after a failed PEAPv2 Part 2 authentication or failed protected termination MUST use other means than successful TLS resumption as the indicator of whether the client is authenticated or not. TLS resumption MUST only be enabled if the implementation supports TLS session cache removal !!!

If an EAP server implementing PEAPv2 removes TLS session cache entries of peers failing PEAPv2 Part 2 authentication, then it MAY skip the PEAPv2 Part 2 conversation entirely after a successful session resumption, successfully terminating the PEAPv2 conversation

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)

Fragementation support is not that easy. Requires sequence numbers ACKs and NAKs (fortunately provided by EAP already), several flags such as (M)ore fragments, (S)tart and a length field.

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**

The input for the cryptographic binding includes the following:

[a] The PEAPv2 tunnel key (TK) is calculated using the first 40 octets of the (secret) key material generated as described in the EAP-TLS algorithm ([RFC2716] Section 3.5). More explicitly, the TK is the first 40 octets of the PRF as defined in [RFC2716]:

PRF(master secret,"client EAP encryption", random)

Where random is the concatenation of client_hello.random and server_hello.random

[b] The first 32 octets of the MSK provided by each successful inner EAP method ;for each successful EAP method completed within the tunnel.

ISK1..ISKn are the MSK portion of the EAP keying material obtained from methods 1 to n. The ISKj shall be the first 32 octets of the generated MSK of the jth EAP method. If the MSK length is less than 32 octets, it shall be padded with 0x00's to ensure the MSK is 32 octets. Similarly, if no keying material is provided for the EAP method, then ISKj shall be set to zero (e.g. 32 octets of 0x00).

The PRF algorithm is based on PRF+ from IKEv2 shown below ("|" denotes concatenation)

K = Key, S = Seed, LEN = output length, represented as binary in a single octet.

PRF (K,S,LEN) = T1 | T2 | T3 | T4 | ... where:

T1 = HMAC-SHA1(K, S | LEN | 0x01)

T2 = HMAC-SHA1 (K, T1 | S | LEN | 0x02)

T3 = HMAC-SHA1 (K, T2 | S | LEN | 0x03)

T4 = HMAC-SHA1 (K, T3 | S | LEN | 0x04)

...

The intermediate combined key is generated after each successful EAP method inside the 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**

The Crypto-Binding TLV MUST be used to perform Cryptographic Binding after each successful EAP method in a sequence of EAP methods is complete in PEAPv2 part 2. The Crypto-Binding TLV can also be used during Protected Termination.

The Crypto-Binding TLV must have the version number received during the PEAP version negotiation. The receiver of the Crypto-Binding TLV must verify that the version in the Crypto-Binding TLV matches the version it sent during the PEAP version negotiation. If this check fails then the TLV is invalid.

The receiver of the Crypto-Binding TLV must verify that the subtype is not set to any value other than the ones allowed. If this check fails then the TLV is invalid.

This message format is used for the Binding Request (B1) and also the Binding Response. This uses TLV type CRYPTO_BINDING_TLV. PEAPv2 implementations MUST support this TLV and this TLV cannot be responded to with a NAK TLV.

The MAC is computed over:

1. The entire Crypto-Binding TLV attribute with the MAC field zeroed out.

2. The EAP Type sent by the other party in the first PEAP message.

3. All the Outer-TLVs from the first PEAP message sent by EAP-server to peer. If a single PEAP message is fragmented into multiple PEAP packets; then the Outer-TLVs in all the fragments of that message MUST be included.

4. All the Outer-TLVs from the first PEAP message sent by the peer to the EAP server. If a single PEAP message is fragmented into multiple PEAP packets, then the Outer-TLVs in all the fragments of that message MUST be included.

EAP-FAST

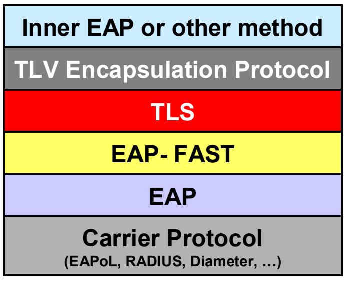
Content

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.

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.

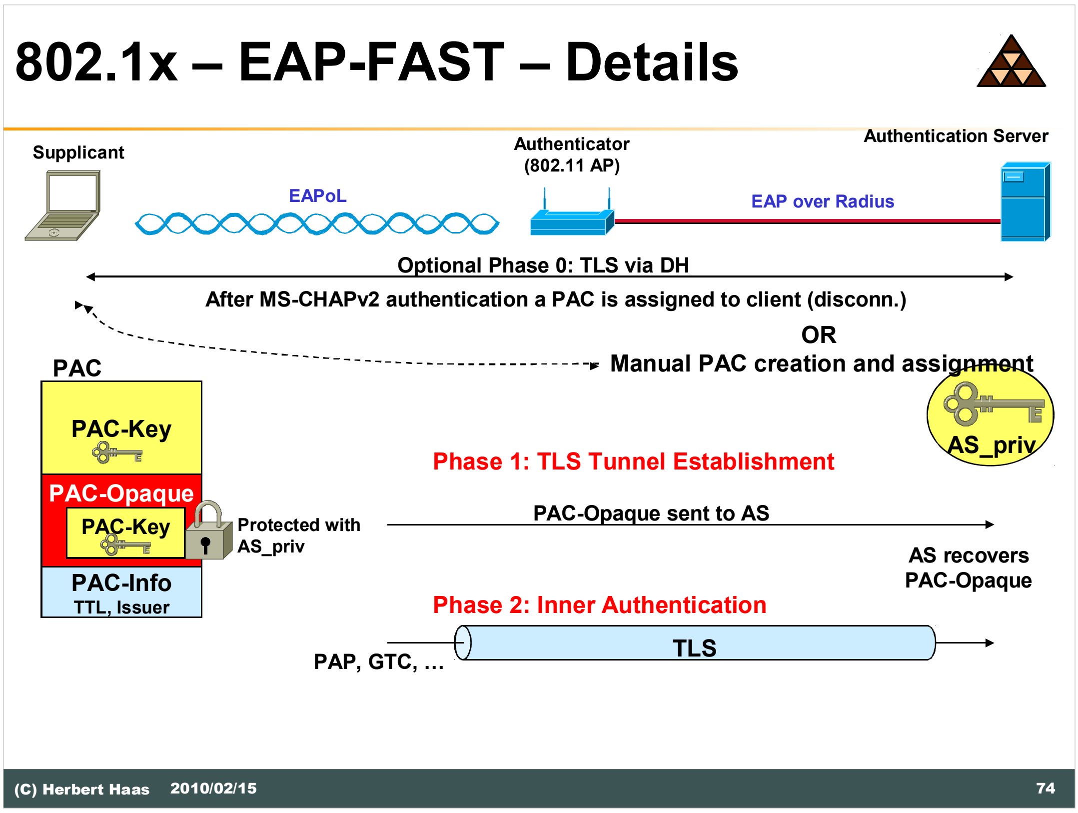
Concept

- 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

(C) Herbert Haas 2010/02/15 73

Today nearly everybody uses Phase 0. Since ACS 4.0 lots of additional EAP-FAST features were introduced including so-called **Authenticated Phase 0** where the server (ACS) is first authenticated using a normal X.509 certificate.

This server certificate is also used to negotiate a tunnel key for Phase 0 (instead of Diffie-Hellman).



As explained in the previous page the optional phase 0 ("PAC provisioning") can be either unauthenticated (DH used) or authenticated (server X.509 certificate used).

Note: Especially when configuring the ACS the keys are named differently:

- The AS_priv key is also known as **Master Key**
- The PAC-key is also known as **Tunnel Key**

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**

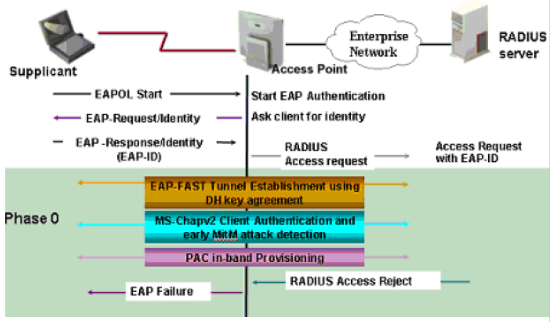
One of the main advantages of EAP-FAST is that authentication servers do not have to maintain state information for each client.

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

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

Manual provisioning ("out-band")

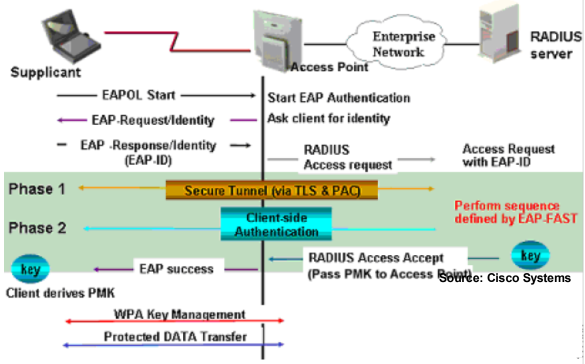
May be necessary if a non-Microsoft-format database is used (such as LDAP) which does not support MSCHAPv2 credentials

PAC files can be manually generated at the ACS and distributed manually to client devices

"Out-of-band" provisioning

EAP-FAST Phases - Detailed

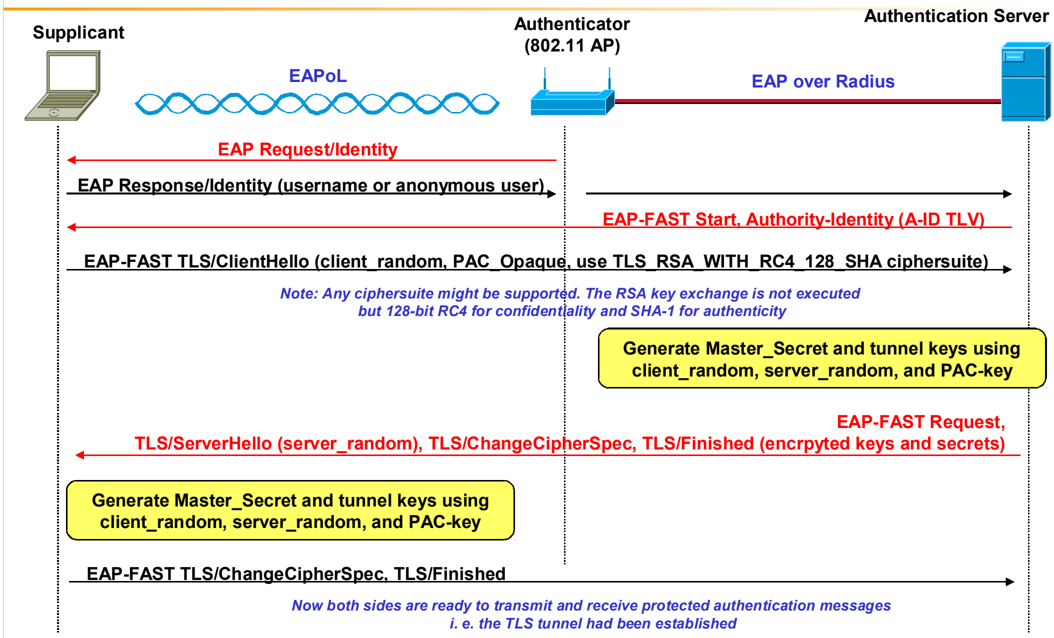
- **Phase 1**
  - Client sends only the PAC opaque to the server, not the PAC key
  - The server decrypts the PAC opaque using its master-key
    - Now server and client have the same PAC key
  - The PAC key is used to create a TLS tunnel for this client's authentication
- **Phase 2**
  - Inside the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



Source: Cisco Systems

The client response is cryptographically bound to the EAP authentication success message. This prevents a Man-In-The-Middle (MITM) attack in which the attacker (client) attempts to provide a false response to the server in order to obtain the session key.

Phase 1 – Details



 78

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]

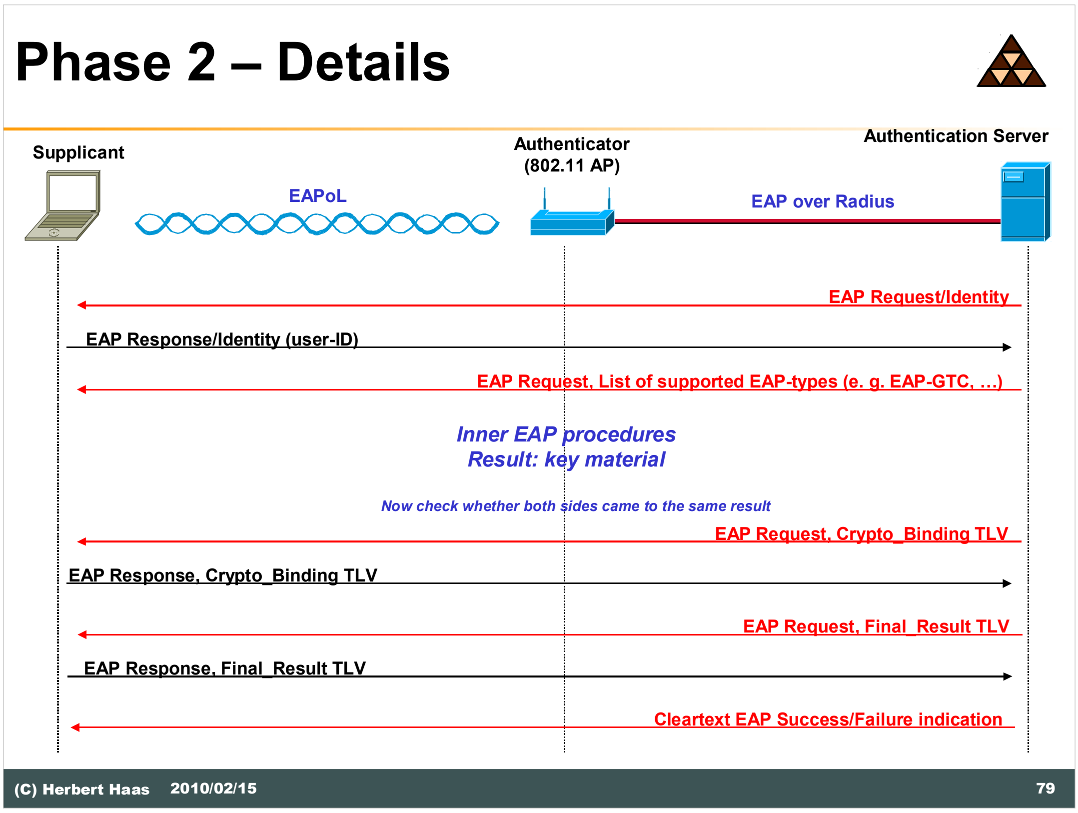
client_write_IV[IV_size=0]

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.



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
  - 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

Content

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

This subchapter provides an overview about the WPA procedures.

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

The basic idea of WPA is to **combine 802.1x authentication with TKIP and MIC.** Furthermore, dynamically established **master keys** should be the basis to calculate dynamic per-user, per-session, and per-packet keys, using TKIP.

Key management can be performed either through RADIUS (like 802.1x is doing, and then it is called "WPA-EAP") or alternatively via **pre-shared keys** without any additional servers. Both mechanisms will generate a master session key for the Authenticator (AP) and Supplicant (client station).

WPA allows to configure "**cipher suites**" on the AP, while the clients may select the most appropriate one during the association process.

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)**

The master key is calculated "pair-wise", that is on the AP as well as ob the client device, either based on 802.1x authentication states or on a Pre-Shared-Key (PSK). The **WPA-PSK** method is only used, when there is no Authentication Server available, typically in home installations.

Note: WPA-PSK is not supported by Cisco ADU or ACU.

Note: When using WPA encryption on an access point, encryption key 1 must not be used as the WPA key negotiation mechanism uses this key position in the AP to transfer authentication data to the client.

WPA2 supports **FIPS 140-2** compliant security, basically AES in counter mode. (An early draft included AES-OCB instead but it was dropped due to patent issues.) A 48 bit IV protects against replay attacks.

Authentication and Integrity is maintained using an **8 byte CBC-MAC** with a 48 bit nonce. Besides the data also the source and destination MAC addresses in the header are protected by the CBC-MAC. (These fields are called Additional Authentication Data (AAD).

The CBC-MAC, the nonce, and additional 2 byte IEEE 802.11 overhead make the CCMP packet 16 octets larger than an unencrypted IEEE 802.11 packet.

The AP advertises cipher suites both in beacons and probe responses.

WPA Concepts

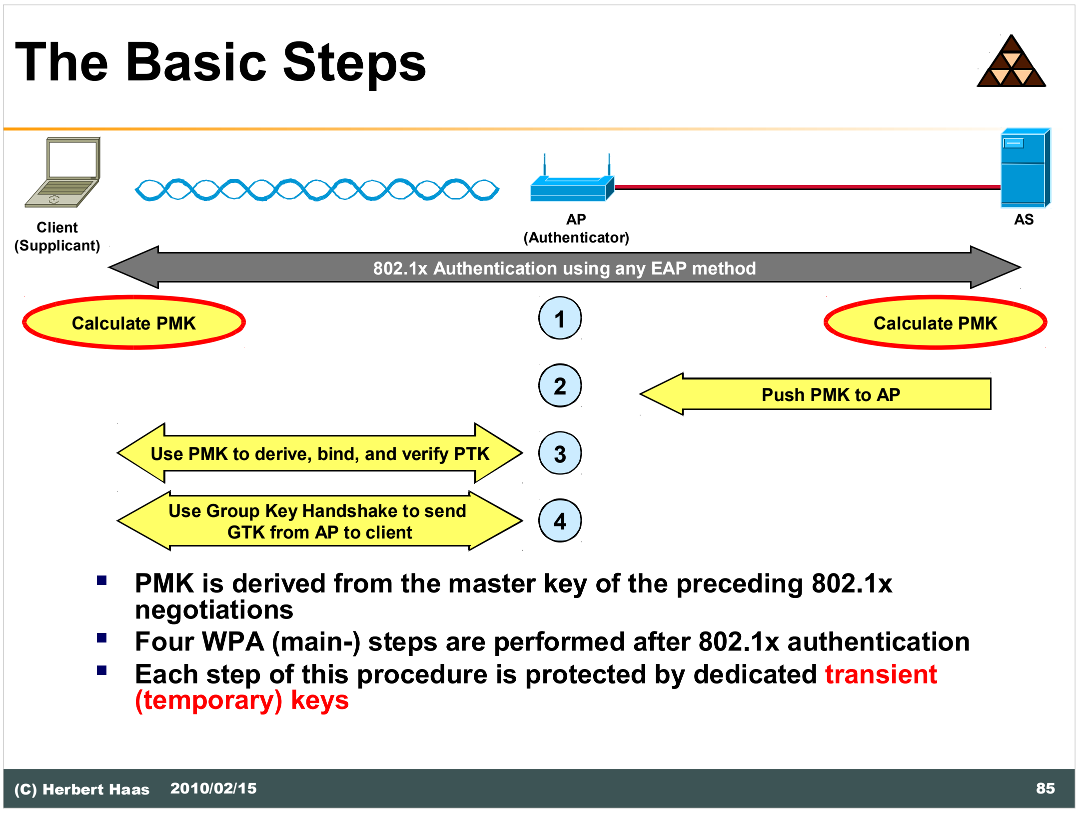
- 1) Pairwise Master Key (PMK) is negotiated between client and AS
  - Based on 802.1x credentials or based on a PSK in home environments
  - PMK is designed to last the entire session
  - Should be exposed as little as possible (therefore PTK needed)
- 2) PMK is pushed from AS to AP
  - Via RADIUS-Access-Accept 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

Unlike WEP, which uses a single key for unicast data encryption and typically a separate key for multicast and broadcast data encryption, WPA uses a set of four different keys for each wireless client-wireless AP pair (known as the pairwise temporal keys) and a set of two different keys for multicast and broadcast traffic.

This set of messages exchanges the values needed to determine the pairwise temporal keys, verifies that each wireless peer has knowledge of the PMK (by verifying the value of the MIC), and indicates that each wireless peer is ready to begin encrypting and providing message integrity protection for subsequent unicast data frames and EAPOL-Key messages.

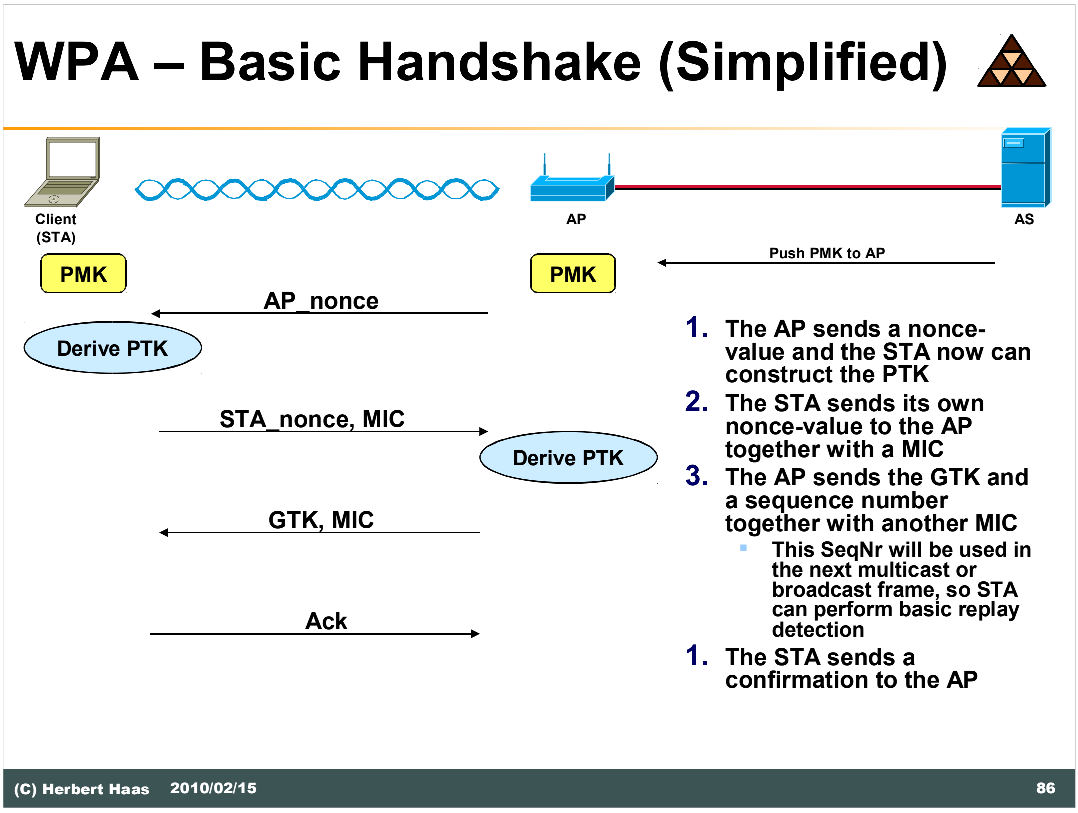
For multicast and broadcast traffic, the wireless AP derives a 128-bit group encryption key and a 128-bit group integrity key and sends these values to the wireless client using an EAPOL-Key message, encrypted with the EAPOL-Key encryption key and integrity-protected with the EAPOL-Key integrity key. The wireless client acknowledges the receipt of the EAPOL-Key message with an EAPOL-Key message.

When a device leaves the network, the GTK also needs to be updated to prevent the device from receiving any more multicast or broadcast messages.



The **Pairwise Master Key (PMK)** is typically calculated using some authentication data which had been derived at the end of a preceding 802.1x/EAP negotiation. For example if EAP-TLS were used, then the PMK = PRF (MasterKey, clientHello.random, serverHello.random, "client EAP encryption")

WPA implements a new 4-Way Handshake and a Group Key Handshake for generating and exchanging data encryption keys between the Authenticator and Supplicant. This handshake is also used to verify that both Authenticator and Supplicant know the master session key.



Note: WPA also includes the requirement to use open key authentication and to obsolete the flawed shared-key authentication. Like 802.11i, WPA capabilities are advertised in beacons, probe responses, association requests, and reassociation requests.

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)

(C) Herbert Haas 2010/02/15 87

Based on the PTK, several temporary working keys are derived:

• A 128-bit **Data Encryption Key** for unicast transmission which is similar as a WEP key and consists of 256-n bits of the PTK key.

• A 128-bit **Data Integrity Key** for unicast MIC

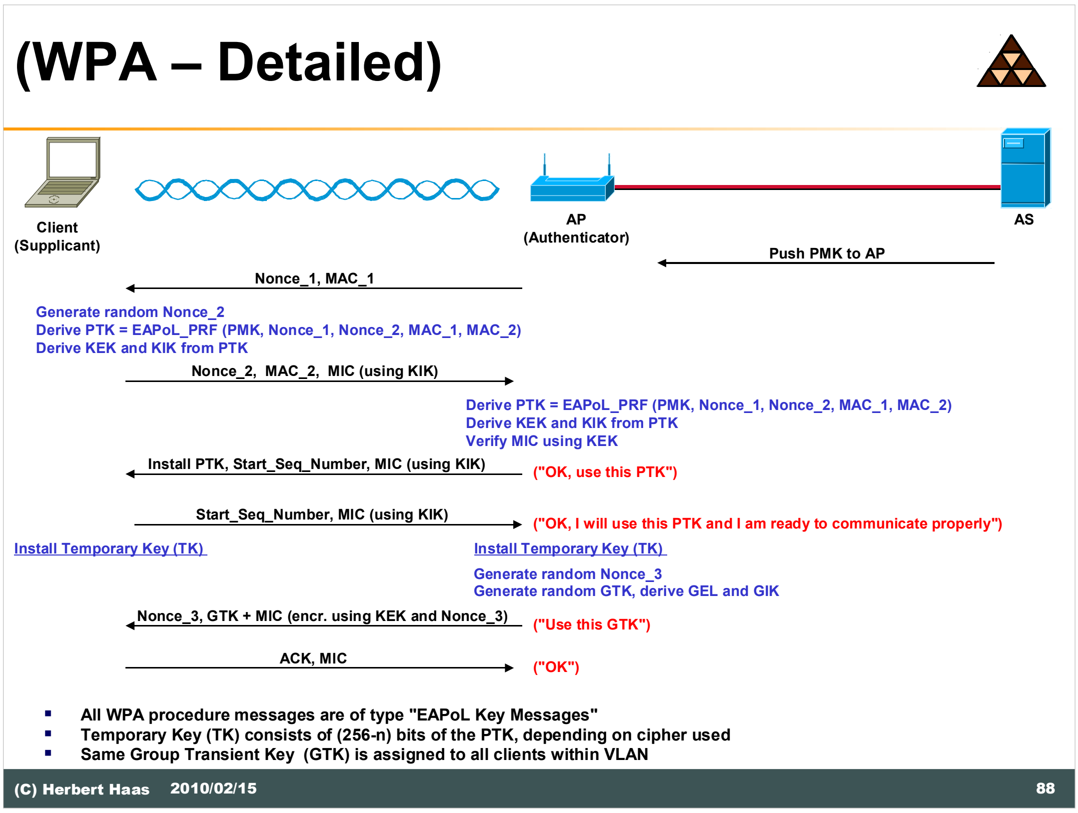
• A 128-bit **EAPoL Key Encryption Key** (KEK) to encrypt EAPoL key messages. This key simply consists of the bits 128-255 of the PTK.

• A 128-bit **EAPoL Key Integrity Key** (KIK) to calculate the MIC for EAPoL key messages. This key is also called "Key Confirmation Key" (KCK) and consists of the bits 0-127 of the PTK.

Based on the GTK, these temporary working keys are derived:

• A 128-bit **Group Encryption Key** (GEK), which is also known as Group Transient Key (GTK) to encrypt multicast and broadcast frames. This key simply consists of the bits 128-255 of the GTK.

•A 128-bit **Group Integrity Key** (GIK) to calculate the MIC for multicast and broadcast frames. This key simply consists of the bits 0-127 of the PTK.



The temporary key exchange is **initiated by the AP** and consists of the following steps:

1. The AP sends an EAPoL key message including Nonce_1 and MAC_1. This message is not encrypted, and no MIC is possible at that stage.
2. Now, the client can calculate a **"Pairwise Transient Key" (PTK)** and derives the KEK and KIK.
3. The client sends an EAPoL key message including Nonce_2 and MAC_2 plus MIC. The MIC is calculated using the EAPoL-KIK.
4. Now, the AP can also derive the PTK and can verify the MIC.
5. The AP sends an EAPoL key message including a MIC and a start-sequence-number to indicate that the AP is now ready to send encrypted unicast frames as well as EAPoL key frames.
6. The client also sends an EAPoL key message including a MIC and a start-sequence-number to indicate that the client is now also ready to send unicast frames as well as EAPoL key frames.
7. The AP finally calculates a 128-bit Group Encryption Key (GEK) as well as a 128-bit Group Integrity Key (GIK) and transmits these values via an EAPoL key message (encrypted with EAPoL-KEK and protected by EAPoL-KIK) to this client.
8. The client acknowledges this message by sending a valid EAPoL key message.

Note: The basic idea of all this is to use a PMK to generate "fresh" PTKs for encryption.

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

According to Microsoft Knowledge Base Article - 815485: "With 802.1X, the rekeying of unicast encryption keys is optional. Additionally, 802.11 and 802.1X provide no mechanism to change the global encryption key used for multicast and broadcast traffic. With WPA, rekeying of both unicast and global encryption keys is required. For the unicast encryption key, the Temporal Key Integrity Protocol (TKIP) changes the key for every frame, and the change is synchronized between the wireless client and the wireless access point (AP). For the global encryption key, WPA includes a facility for the wireless AP to advertise the changed key to the connected wireless clients."

WPA-2 PMK Caching: **PKC** allows a client to store PMKs to reuse them when later associated to the same AP or LAP. In order to support PKC the clients calculates and sends PMKIDs, i. e. a hash of the PMK, a string, the station MAC and the AP MAC. This 'PMK SA Identifier' is sent in an association request. The PMKID uniquely identifies the PMK on the WLC and therefore the 802.1x authentication can be by-passed. The client can send more than one key name in the association request. If the access point or WLC sends a success in the association response, then the client and access point proceed directly to the 4-way handshake.

Note:

- PKC is automatically enabled on a Cisco WLC when WPA2 is enabled for a WLAN.
- PKC does not work with Aironet Desktop Utility (ADU) as client supplicant.
- PMK cache records kept for 1 hour for non associated clients.

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

While PKC reduces the reauthentication time on APs or WLCs where the client has been authenticated once, preauthentication reduces roaming delays because it allows clients to authenticate to other APs or WLCs without association. Note that the preauthentication process is realized through the current AP or WLC to which the client is currently associated! Using preauthentication the client can establish PMKs with all APs or WLCs. The PTK handshake is only performed when the client actively associates to a new AP or WLC. In this case the association request again carries a PMK SA Identifier as explained in the PKC section above.

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

The alternative to server-based keys (SBKs). In WPA-PSK, users must share a passphrase that may be from eight to 63 ASCII characters or 64 hexadecimal digits (256 bits). Each character in the pass-phrase must have an encoding in the range of 32 to 126 (decimal), inclusive. (IEEE Std. 802.11i-2004, Annex H.4.1). The space character is included in this range.

In November 2003, Robert Moskowitz, a senior technical director at ICSA Labs (part of TruSecure) released "**Weakness in Passphrase Choice in WPA Interface**". In this paper, Moskowitz described a straightforward formula that would reveal the passphrase by performing a dictionary attack against WPA-PSK networks. This weakness is based on the fact that the pairwise master key (PMK) is derived from the combination of the passphrase, SSID, length of the SSID and nonces. The concatenated string of this information is hashed 4,096 times to generate a 256-bit value and combine with nonce values. The information required to create and verify the session key is broadcast with normal traffic and is readily obtainable; the challenge then becomes the reconstruction of the original values. Moskowitz explains that the pairwise transient key (PTK) is a keyed-HMAC function based on the PMK; by capturing the four-way authentication handshake, the attacker has the data required to subject the passphrase to a dictionary attack.

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, …

According to Moskowitz, "a key generated from a passphrase of less than about 20 characters is unlikely to deter attacks." In late 2004, Takehiro Takahashi, then a student at Georgia Tech, released **WPA Cracker**.

Around the same time, Josh Wright, a network engineer and well-known security lecturer, released **coWPAtty**. Both tools are written for Linux systems and perform a brute-force dictionary attack against WPA-PSK networks in an attempt to determine the shared passphrase. Both require the user to supply a dictionary file and a dump file that contains the WPA-PSK four-way handshake. Both function similarly; however, coWPAtty contains an automatic parser while WPA Cracker requires the user to perform a manual string extraction.

Additionally, coWPAtty has optimized the HMAC-SHA1 function and is somewhat faster. Each tool uses the PBKDF2 (Password-Based Key Derivation Function) algorithm that governs PSK hashing to attack and determine the passphrase. Neither is extremely fast or effective against larger passphrases, though, as each must perform 4,096 HMAC-SHA1 iterations with the values as described in the Moskowitz paper.

PBKDF2 is a key derivation function that is part of RSA Laboratories' Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 v2.0, also published as Internet Engineering Task Force's RFC 2898. It replaces an earlier standard, PBKDF1, which could only produce derived keys up to 160 bits long.