

# *Communications Services and Security* **Wireless Networks**

Cèsar Fernández

Departament d'Informàtica  
Universitat de Lleida

Curs 2022 - 2023



## 1 Wireless Ethernet

- WLAN Technologies
- Configurations
- Medium Access Control (MAC)
- WLAN Frames
- IEEE 802.11 Extensions

## 2 WiFi Security

- Contents outline

- WiFi connection
- Authentication
- WEP, Wireless Enhanced Privacy
- 802.11i

## 3 Wifi deployments

- Centralized control architecture
- Alcatel-Lucent Instant AP

## 4 Bibliography



# Contents

- 1 **Wireless Ethernet**
  - WLAN Technologies
  - Configurations
  - Medium Access Control (MAC)
  - WLAN Frames
  - IEEE 802.11 Extensions
- 2 WiFi Security
- 3 Wifi deployments
- 4 Bibliography



# Wireless Ethernet

## Introduction

- **WLAN** (Wireless Local Area Network) rises as an alternative to wired networks
- Two main motivations:
  - Sometimes, wired is not possible
  - Equipment mobility requirements
- To employ radio frequency
- Range station determined by radio equipment power
- At 1.990 IEEE starts the **802.11** working group



# WLAN Technologies

## Two main transmission techniques

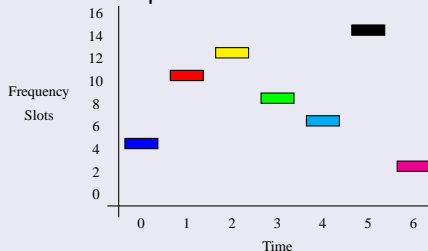
- **Narrow Band**
  - A frequency band is assigned to each client
  - Frequency assignment such that **no interferences** among clients
- **Broad Band**
  - Clients physically interfere among them
  - **Spread spectrum** techniques recover interfering situation
  - So, more immunity to external interfering sources

# WLAN Technologies

## Two main Spread Spectrum techniques

### FHSS (Frequency-Hopping Spread Spectrum)

- The **carrier frequency** changes (hops) frequently (several times per second)
- Changing patterns known by transmitter and receiver. Synchronization required
- Any non-synchronized reception is seen as **noise**

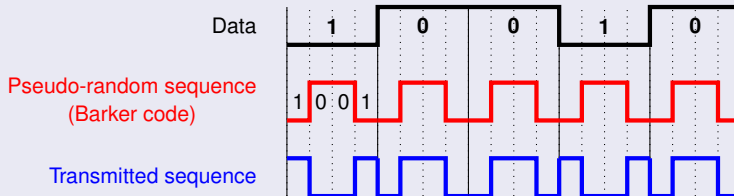


# WLAN Technologies

## Two main Spread Spectrum techniques

### DSSS (Direct-Sequence Spread Spectrum)

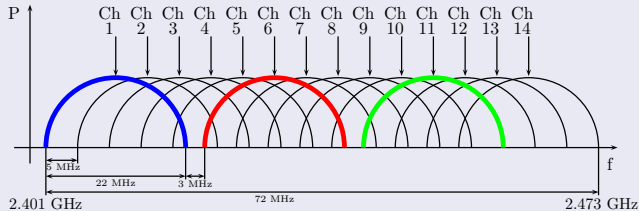
- Each information bit **XORed** by a faster bit sequence (**chips**)
- Chip sequences are **pseudo-random** and **orthogonal**
- As in frequency hopping, chip sequences are agreed by pairs (**synchronized**)
- Any non-synchronized reception is seen as **noise**



- Such a faster chip sequence makes broader the transmission band

# WLAN Technologies

## Chanel distribution



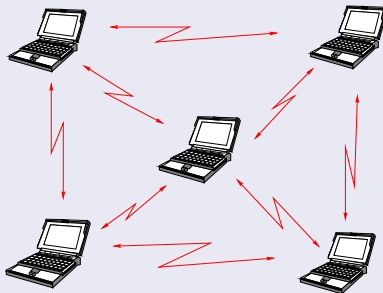


# Configurations

## Two type of WLAN configurations

### Simple networks (*Ad-hoc*)

Client communication is allowed for stations inside the cover range

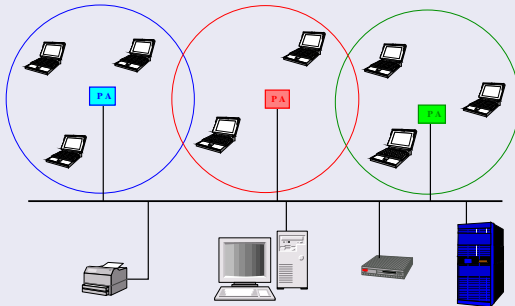


# Configurations

## Two type of WLAN configurations

### Distributed networks (*Infrastructure*)

- A wired infrastructure exists where **Access Points** are attached
- Access points serve a set of mobile clients, giving a **cell** coverage



# Medium Access Control (MAC)

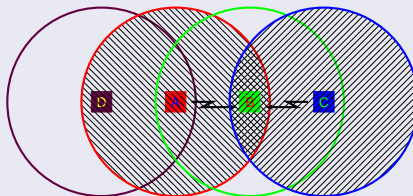
## Some problems

- To employ CSMA for WLAN is **inefficient**
- Because of limited range, CSMA presents two problems:
  - **Hidden station**
  - **Exposed station**

# Medium Access Control (MAC)

## Hidden station problem

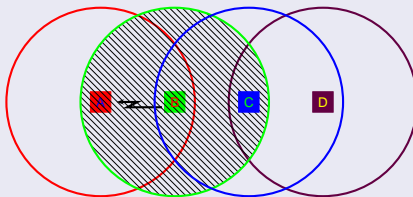
- A sends to B
- C senses the medium and detects it free (because out of range). C starts sending
- C interferes B, masking A reception



# Medium Access Control (MAC)

## Exposed station problem

- B sends to A
- C senses the medium busy. Can not send to D, even possible because out of range from A



# Medium Access Control (MAC)

## MACA protocol (Multiple Access with Collision Avoidance)

Early protocol for WLAN (Karn, 1990)

Procedure:

- When A wants to send to B, A sends a **RTS** (*Request To Send*) to B. The RTS frame contains the **size** of the data to be sent
- All the stations receiving RTS from A (inside A range) keep quiet until B responds
- B sends a **CTS** (*Clear To Send*) to A, telling its availability to receive data. CTS frame also have the data size from A
- All the stations receiving CTS from B (inside B range) keep quiet during B transmission time
- When a RTS is sent and no response is received, enters into a **exponential binary backoff** retransmission



# Medium Access Control (MAC)

## MACAW protocol (Multiple Access with Collision Avoidance for Wireless)

Modified MACA based (Bharghavan, 1.994). Efficiency improved

- Receiving station sends **ACK** when a frame has been correctly received
- To avoid initial collisions CSMA is employed for RTS frames
- Exponential binary backoff employed for each connection instead of for each station



# WLAN Frames

## WLAN frame types

- 1 **Data frames**: End user information frames
- 2 **Control frames**: MAC operation related frames, such as RTS, CTS, ACK, ...
- 3 **Management frames**: Management operations related to beacons, authentication, SSID, ...



# WLAN Frames

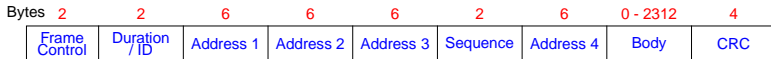
## WLAN frame structure

It is common for all WLAN frames



- **Preamble**: two fields
  - *Sync*: 80 bits (0101...) for receiver synchronization
  - *SDF* (*Start Frame Delimiter*): 16 bits (0000 1100 1011 1101)
- **PLCP Header** (*Physical Layer Convergence Procedure*):  
Information to decode the MAC frame
  - *Signaling*: information rate (8 bits)
  - *Service*: modulation type (8 bits)
  - *Length*: frame length (16 bits)
  - *CRC*: 16 bits for error detection
- **MAC Frame**: Layer 2 frame

# WLAN Frames: The MAC frame



MAC Frame Format

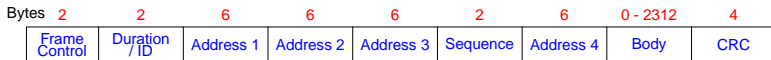
## Frame control



- **Protocol**: Set to 00
- **Type**:
  - **00**: Management (Association req/res, Probe req/res, Authentication, Beacon, ...)
  - **01**: Control (RTS, CTS, ACK, ...)
  - **10**: Data
- **Subtype**:
- **To DS**: Set to 1 for frames going to AP
- **From DS**: Set to 1 for frames coming from AP



# WLAN Frames: The MAC frame



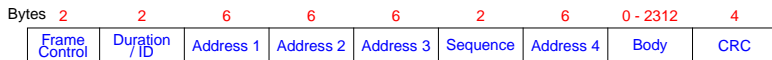
MAC Frame Format

## Frame control



- **Fragmentation**: indicates fragmented data
- **Retransmission**: set to 1 if frame has been retransmitted
- **PM**: (*Power Management*) tells that station is going to enter in power saving mode
- **PMD**: set to 1 if AP has pending data for a PM mode station
- **WEP**: enciphered data
- **Order**: for protocols DEC and LAT only

# WLAN Frames: The MAC frame



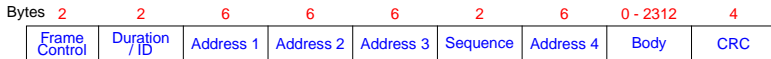
MAC Frame Format

## Duration / ID

Its value depends on the frame type:

- For stations in PM mode indicates the association ID (AID). So, when station wakes up, AP sends the corresponding buffered data
- Otherwise, indicates the time that channel is expected to be busy during the current transmission (in microseconds)

# WLAN Frames: The MAC frame



MAC Frame Format

## Addresses

Their content depends on **To DS** and **From DS** values

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	Destination	Source	BSSID	N / A <sup>a</sup>
0	1	Destination	BSSID	Source	N / A
1	0	BSSID	Source	Destination	N / A
1	1	AP <sub>rx</sub>	AP <sub>tx</sub>	Source	Destination <sup>b</sup>

<sup>a</sup>Ad-Hoc networks, Control and Management frames

<sup>b</sup>Wireless bridges

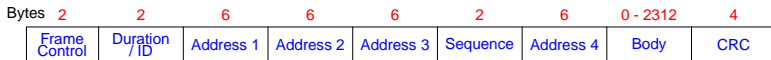
## BSSID (Basic Service Set Id)

BSSID allows to differentiate WLAN in the same area.

- For infrastructure networks, BSSID is the AP MAC wired address
- For adhoc networks, BSSID is randomly generate



# WLAN Frames: The MAC frame



MAC Frame Format

## Addresses

Their content depends on **To DS** and **From DS** values

To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	Destination	Source	BSSID	N / A <sup>a</sup>
0	1	Destination	BSSID	Source	N / A
1	0	BSSID	Source	Destination	N / A
1	1	$AP_{rx}$	$AP_{tx}$	Source	Destination <sup>b</sup>

<sup>a</sup>Ad-Hoc networks, Control and Management frames

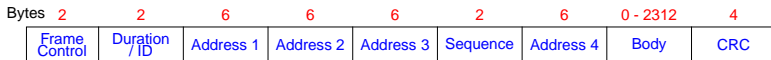
<sup>b</sup>Wireless bridges

## ESSID (Extended SSID) (or SSID for short)

- ESSID is a set of 32 ASCII chars. Included in management frames (beacon, ...)
- Consists of all the BSSID of the network



# WLAN Frames: The MAC frame



MAC Frame Format

## Sequence

For fragmented frames indicates their sequence order

## CRC

Cyclic Redundancy Check

# IEEE 802.11 Extensions

- New features and specifications based on IEEE 802.11
- Higher transmission rates for WLAN
- Only physical layer is modified
- New modulation techniques to enhance throughput





# IEEE 802.11 Extensions: 802.11b

## 802.11b

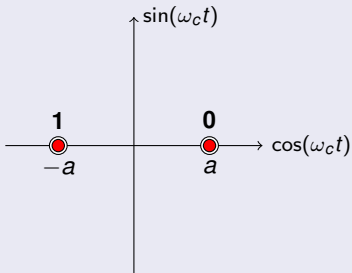
- Operates at 2.4 GHz, ISM band (*Industrial Scientific and Medical*), known as "Wi-Fi" (*Wireless Fidelity*).
- Employs DSSS at transmission rates of 1, 2, 5.5 and 11 Mbps.
- Barker Code signal being modulated as QPSK (*Quadrature Phase Shift Keying*) at 2, 5.5 i 11 Mbps and as BPSK (*Binary Phase Shift Keying*) at 1 Mbps.
- 802.11b backward 802.11 compatible at 1 and 2 Mbps.



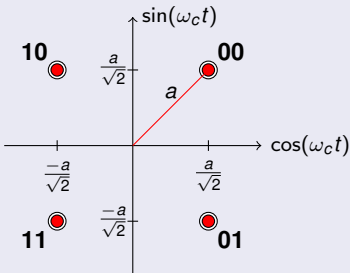
# IEEE 802.11 Extensions: 802.11b

## Modulations

### BPSK



### QAM



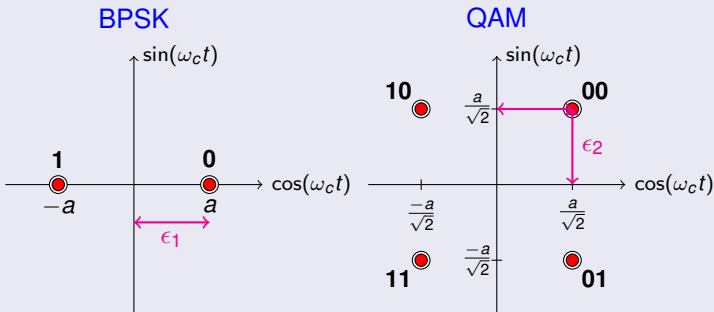
$$10_{bpsk} = \begin{cases} -a \cdot \cos(\omega_c t), & 0 \leq t < T_s, \quad (T_s = T_b) \\ a \cdot \cos(\omega_c(t - T_s)), & T_s \leq t < 2T_s \end{cases}$$

$$10_{qam} = -\frac{a}{\sqrt{2}} \cdot \cos(\omega_c t) + \frac{a}{\sqrt{2}} \cdot \sin(\omega_c t), \quad 0 \leq t < T_s, \quad (T_s = 2T_b)$$



# IEEE 802.11 Extensions: 802.11b

## Modulations

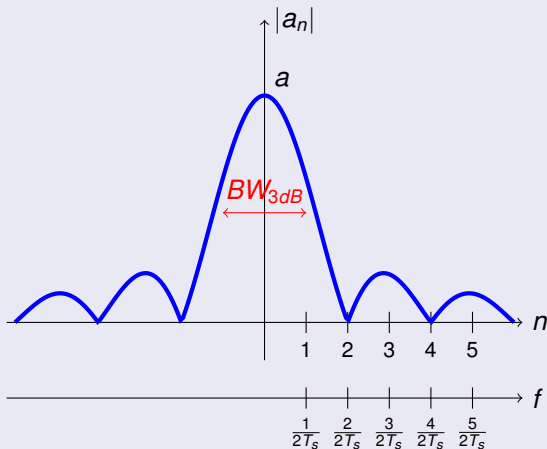


- QAM double transmission rate with the same bandwidth
- QAM more error sensitivity at the same power strength

# IEEE 802.11 Extensions: 802.11b

## Bandwidth for a $2T_s$ period pulse

$$a_n = \frac{4a}{\pi n} \sin(\pi n/2)$$

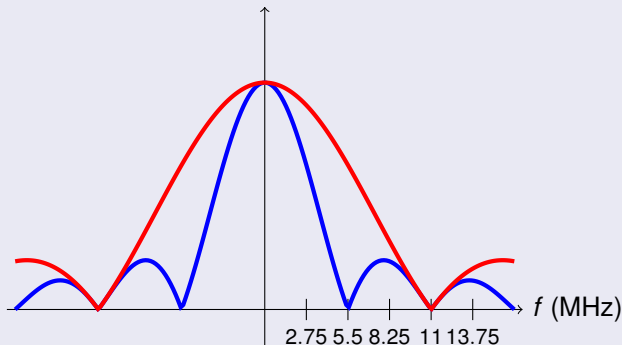


# IEEE 802.11 Extensions: 802.11b

## Bandwidth for a $2T_s$ period pulse

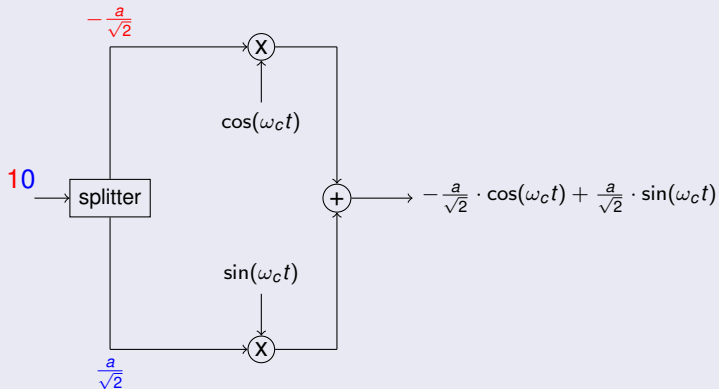
At 11 Mbps:

- QAM,  $T_s = 0.18 \mu\text{s}$
- BPSK,  $T_s = 0.09 \mu\text{s}$



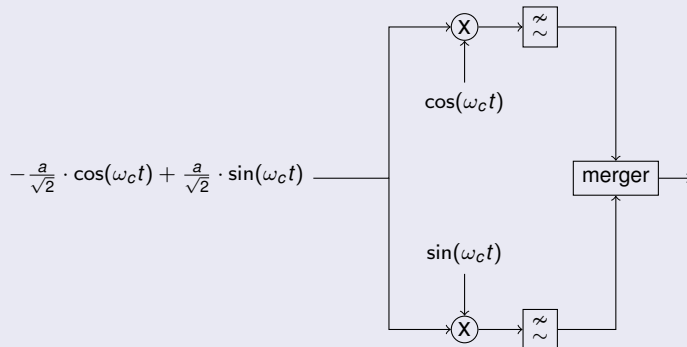
# IEEE 802.11 Extensions: 802.11b

## Transmission and reception outline



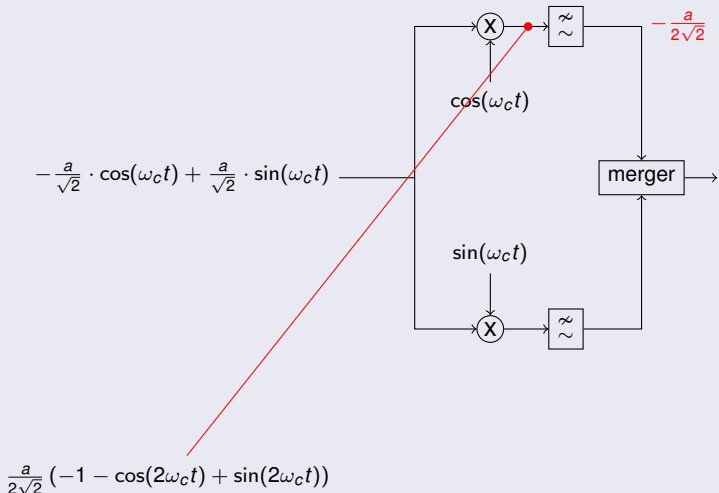
# IEEE 802.11 Extensions: 802.11b

## Transmission and reception outline



# IEEE 802.11 Extensions: 802.11b

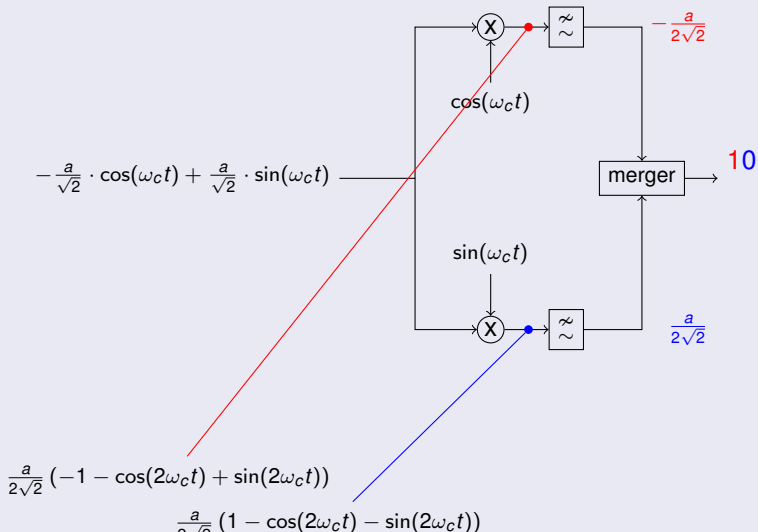
## Transmission and reception outline





# IEEE 802.11 Extensions: 802.11b

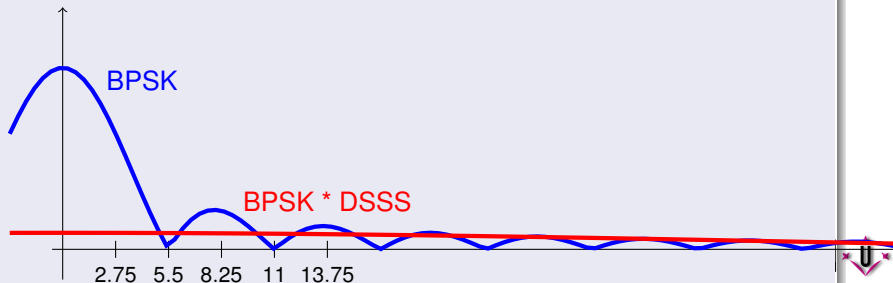
## Transmission and reception outline



# IEEE 802.11 Extensions: 802.11b

## CDMA - DSSS

- Each information bit **XORed** with an established Barker Code
- Barker code consists on 11 chips for BPSK: + - + + - + + - - -
- Multiple access technique as a modulation engine
- Deals better against interferences; orthogonality
- Spread Spectrum



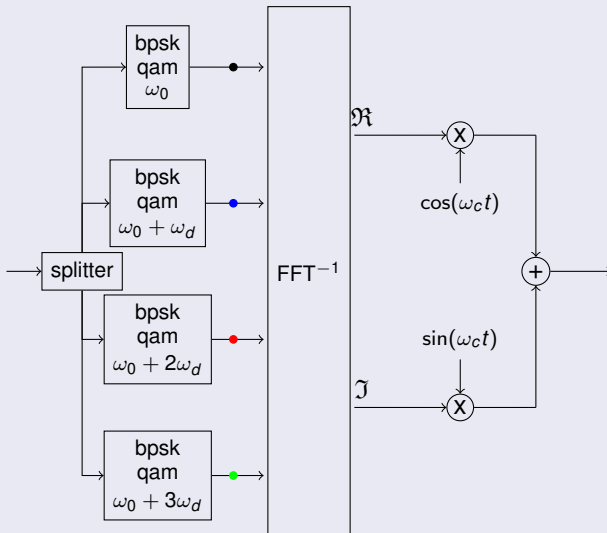
# IEEE 802.11 Extensions: 802.11a

## 802.11a

- Working at **5 GHz** (USA: 5.18 - 5.32, 5.745-5.805 GHz). UNII band (*Unlicensed National Information Infrastructure*)
- 33 channels (UNII-1 + UNII-2A + UNII-2C). 20 MHz separation
- Non compatible with the rest of wireless networks (working at 2.4 GHz.)
- Modulation **OFDM** (*Orthogonal Frequency Division Multiplexing*)  
 $V_{T_{Max}}$  **54 Mbps**.
- OFDM splits a high speed carrier into 52 **orthogonal** sub-carriers, transmitting on the 52 sub-carriers at the same time
- Available rates: **6, 9, 12, 18, 24, 36, 48** and **54 Mbps**

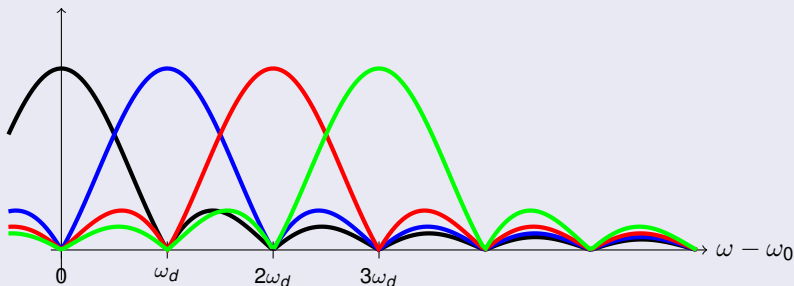
# IEEE 802.11 Extensions: 802.11a

## Example: 4 sub-carriers transmitter



# IEEE 802.11 Extensions: 802.11a

## Example: 4 sub-carriers transmitter



# IEEE 802.11 Extensions

## 802.11g

- Same rates than 802.11a but 802.11b backward compatibility
- Operates at 2.4 GHz in ISM band
- For high transmission rates, OFDM is used, as in 802.11a
- 802.11g devices automatically change to QPSK modulation if only 802.11b compliant devices are detected



# IEEE 802.11 Extensions

## 802.11n

- Transmission rates up to 600 Mbps
- a/b/g interoperability
- Several technologies employed
  - **Multiple Input Multiple Output (MIMO)**
  - **Channel Bonding**
  - MAC protocol enhancements

# IEEE 802.11 Extensions

## 802.11ac

- Fifth generation wifi
- Transmission rates up to 6.9 Gbps
- Only operates at 5 GHz band
- Evolution of 802.11n
- Pushing forward 802.11n technologies (MIMO, Modulation)





# IEEE 802.11 Extensions: 802.11n

## MIMO

- Devices (APs and clients 802.11n) with multiple antennas



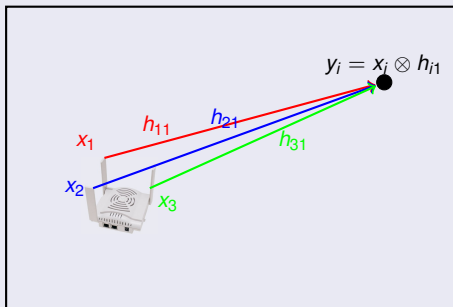
- Notation  $M \times N$  indicates:  $M$  transmitting and  $N$  receiving antennas
- AP on picture may operate with the following configurations:  $3 \times 3$ ,  $3 \times 2$ ,  $2 \times 3$ , ...
- $M \geq 2$  for APs.  $M \geq 1$  for clients



# IEEE 802.11 Extensions: 802.11n

## MIMO

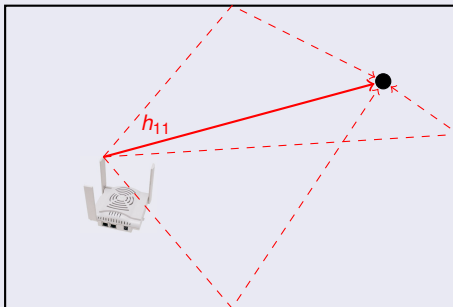
- The transmitter split the information into  $M$  streams, being transmitted at the same time through each antenna **Spatial Multiplex**
- Each path **spatial signature** allows discrimination, determined by the **multipath** effect



# IEEE 802.11 Extensions: 802.11n

## MIMO

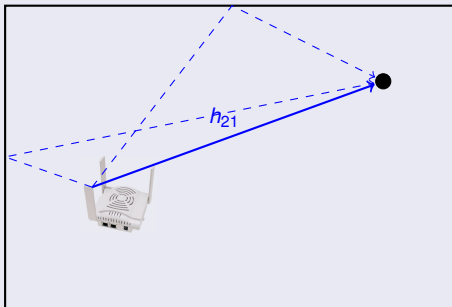
- The transmitter split the information into  $M$  streams, being transmitted at the same time through each antenna **Spatial Multiplex**
- Each path **spatial signature** allows discrimination, determined by the **multipath** effect



# IEEE 802.11 Extensions: 802.11n

## MIMO

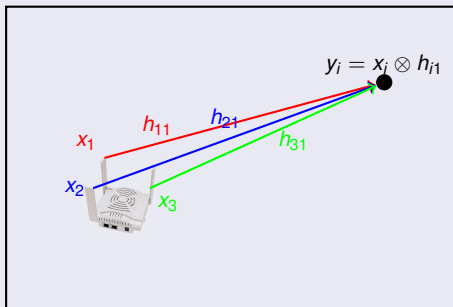
- The transmitter split the information into  $M$  streams, being transmitted at the same time through each antenna **Spatial Multiplex**
- Each path **spatial signature** allows discrimination, determined by the **multipath** effect



# IEEE 802.11 Extensions: 802.11n

## MIMO

- The transmitter split the information into  $M$  streams, being transmitted at the same time through each antenna **Spatial Multiplex**
- Each path **spatial signature** allows discrimination, determined by the **multipath** effect



Multipath helps MIMO !



# IEEE 802.11 Extensions: 802.11n

## Channel Bonding

- Two channels are bonded, on each band (ISM or UNII; b/g or a)
- 20 MHz each channel
- 40 MHz total bandwidth



# IEEE 802.11 Extensions: 802.11n

## Modulation and Coding Scheme (MCS)

MCS	Type	<i>Coding rate</i>	SS	Mbps (20 MHz)	Mbps (40 MHz)
0	bpsk	1/2	1	6.5	13.5
1	qpsk	1/2	1	13.	27.
2	qpsk	3/4	1	19.5	40.5
...	...	...	...	.....	.....
7	64-qam	5/6	1	65.	135.
8	bpsk	1/2	2	13.	27.
...	...	...	...	.....	.....
15	64-qam	5/6	2	130.	270.
16	bpsk	1/2	3	19.5	40.5
...	...	...	...	.....	.....
31	64-qam	5/6	4	260.	540.

# IEEE 802.11 Extensions: 802.11n

## Modulation and Coding Scheme (MCS)

Modulation and coding schemes							
MCS Index	Spatial streams	Modulation type	Coding rate	Data rate (in Mbit/s) <sup>[a]</sup>			
				20 MHz channel		40 MHz channel	
				800 ns GI	400 ns GI	800 ns GI	400 ns GI
0	1	BPSK	1/2	6.5	7.2	13.5	15
1	1	QPSK	1/2	13	14.4	27	30
2	1	QPSK	3/4	19.5	21.7	40.5	45
3	1	16-QAM	1/2	26	28.9	54	60
4	1	16-QAM	3/4	39	43.3	81	90
5	1	64-QAM	2/3	52	57.8	108	120
6	1	64-QAM	3/4	58.5	65	121.5	135
7	1	64-QAM	5/6	65	72.2	135	150
8	2	BPSK	1/2	13	14.4	27	30
9	2	QPSK	1/2	26	28.9	54	60
10	2	QPSK	3/4	39	43.3	81	90
11	2	16-QAM	1/2	52	57.8	108	120
12	2	16-QAM	3/4	78	86.7	162	180
13	2	64-QAM	2/3	104	115.6	216	240

14	2	64-QAM	3/4	117	130	243	270
15	2	64-QAM	5/6	130	144.4	270	300
16	3	BPSK	1/2	19.5	21.7	40.5	45
17	3	QPSK	1/2	39	43.3	81	90
18	3	QPSK	3/4	58.5	65	121.5	135
19	3	16-QAM	1/2	78	86.7	162	180
20	3	16-QAM	3/4	117	130	243	270
21	3	64-QAM	2/3	156	173.3	324	360
22	3	64-QAM	3/4	175.5	195	364.5	405
23	3	64-QAM	5/6	195	216.7	405	450
24	4	BPSK	1/2	26	28.8	54	60
25	4	QPSK	1/2	52	57.6	108	120
26	4	QPSK	3/4	78	86.8	162	180
27	4	16-QAM	1/2	104	115.6	216	240
28	4	16-QAM	3/4	156	173.2	324	360
29	4	64-QAM	2/3	208	231.2	432	480
30	4	64-QAM	3/4	234	260	486	540
31	4	64-QAM	5/6	260	288.8	540	600

From wikipedia





# IEEE 802.11 Extensions: 802.11n

## Enhancement Factors

Rate	Technique
54 Mbps	OFDM conventional (48 subcarriers) As 802.11g/802.11a
65 Mbps	Up to 52 subcarriers
135 Mbps	Channel bonding (40 MHz)
< 600 Mbps	Up to MIMO 4×4



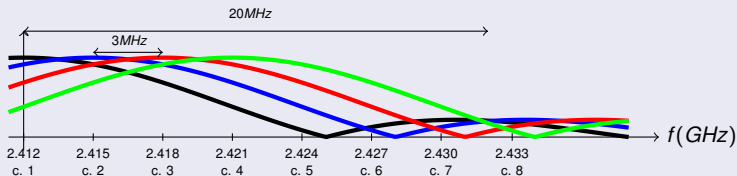
# IEEE 802.11 Extensions: 802.11n

## Comparing channel assignment for ISM and UNII bands

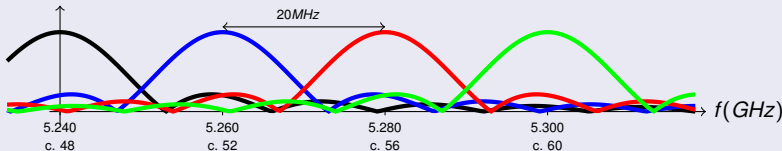
MCS=0 (bspk, 6.5 Mbps, Coding rate=1/2). Bits=101010...  $T_0 = 1/6.5\mu s = 0.15\mu s$ .

$BW_{between\_zeros} = 26 \text{ MHz}$

### ISM



### UNII



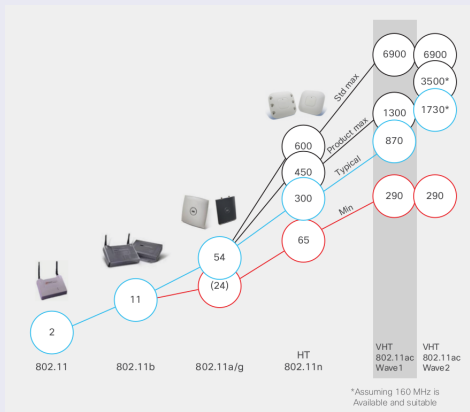
# IEEE 802.11 Extensions: 802.11ac

## Enhancements

- Modulations up to 256 QAM with  $1/r = 5/6$
- More spatial streams,  $M = 8$
- Larger bandwidths: 20, 40 , 80 and 160 MHz

# IEEE 802.11 Extensions: 802.11ac

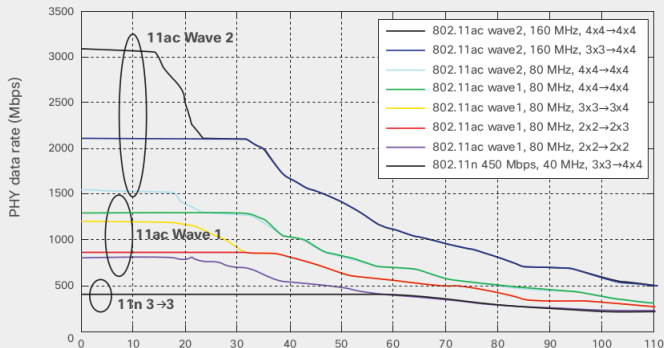
## 802.11 bandwidth evolution



From: 802.11ac: The Fifth Generation of Wi-Fi Technical White Paper. Cisco

# IEEE 802.11 Extensions: 802.11ac

## 802.11 bandwidth range



From: 802.11ac: The Fifth Generation of Wi-Fi Technical White Paper. Cisco

# Contents

## 1 Wireless Ethernet

## 2 WiFi Security

- Contents outline
- WiFi connection
- Authentication
- WEP, Wireless Enhanced Privacy
- 802.11i

## 3 Wifi deployments

## 4 Bibliography



# Contents outline

- **Connection process.** Scan, authentication and association
- **Authentication**
  - Mechanisms (SSID, MAC, Captive Portal, 802.1x)
  - Protocols (PAP, CHAP, MSCHAP, EAP)
  - 802.1x (EAPOL)
- **Ciphers**
  - WEP (outdated)
  - 802.11i (WPA2)
    - TLS (Protocol for authentication and key agreement)
    - TKIP (Protocol for key interchange)



# WiFi connection

## Connection process phases

- 1 Scan
- 2 Authentication
- 3 Association



# WiFi connection

## Scan

- Client searches for a compatible wireless network
- **Beacon frames** employed
- Two working modes:
  - **Active scan.** Client sends **Probe Request** frames through each available channel
  - **Passive scan.** Client doesn't transmit. Expecting beacon frames

The scan scope may be limited to a given **SSID**

# WiFi connection

## Authentication

- Directed authentication. Clients → network
- Employed frames: Authentication Request, Authentication Response
- Two mechanisms:
  - 1 Open System Authentication (OSA). Default mechanism
  - 2 Shared Key Authentication (SKA).  
Employs **Wired Equivalent Privacy** (WEP) protocol. Both, AP and client shares a common key

Upon both mechanisms, manufacturers use proprietary authentication methods (based on MAC address, on SSID, Captive Portal, ...)



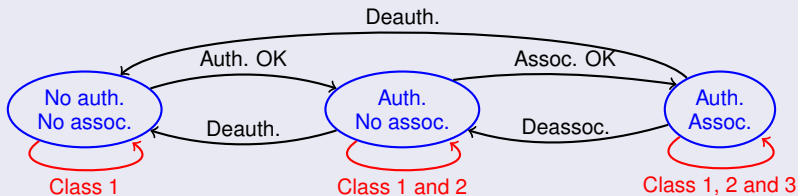
# WiFi connection

## Association

- Employed frames: [Association Request](#) and [Association Response](#)
- When client receives [Association Response](#), WiFi services are operative

# WiFi connection

## Connection states



Class	Control	Management	Data
1	RTS CTS ACK	Probe Request Probe Response Beacon Authentication Deauthentication ATIM	To DS = 0 and From DS = 0
2		Association Request/Request Reassociation Request/Request Disassociation	
3		Deauthentication	To DS = 1 or From DS = 1

# WiFi connection

## Access Control Lists (ACL)

- ACL not included in 802.11 standard, but implemented in most APs
- It is a list of allowed MAC
- During **association**, the client MAC is checked in ACL
- If check fails, *Association Response* is sent, denying its access



# Authentication

## Methods

- Based on SSID (Service Set Identifier)
- Based on MAC
- Captive Portal
- 802.1x

## Based on SSID

- Used on a first stage (open or protected) to allocate the clients at the corresponding subnet (corp, visitor, ...)
- Then, proceed with the corresponding authentication method, i.e. (corp: 802.1x, visitor: Captive Portal)

# Authentication

## Based on MAC

- Insecure
- Suitable for light devices with few ability to authenticate (printers, scanners, ...)
- MAC tables must be defined at APs or WiFi Controllers (centralized deployments)

# Authentication

## Captive Portal

- Open Authentication/Association
- Useful when changing users (hotels, visitors, ...)
- Steps:
  - 1 Client associates to an open SSID. Gets IP, DNS, ...
  - 2 Firewall (AP or central) allows DNS queries. Drops any other traffic



- 3 Client HTTP requests are NAT-Destined to an authentication web (via https)
- 4 Once authenticated, the firewall opens traffic for the client and sends a **HTTP redirect** to the client





# Authentication

## Authentication protocols

- PAP (*Password Authentication Protocol*)
- CHAP (*Challenge Authentication Protocol*)
- MSCHAP (*Microsoft Challenge Authentication Protocol*)
- EAP (*Extensible Authentication Protocol*)

# Authentication

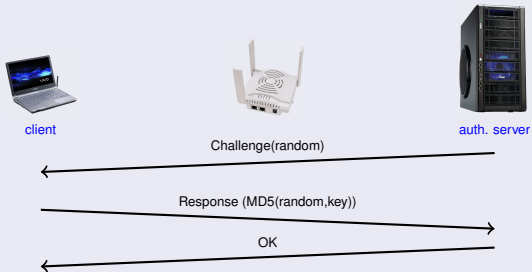
## PAP (*Password Authentication Protocol*)

- The client sends `login:password` in clear text
- Insecure.

# Authentication

## CHAP (Challenge Authentication Protocol)

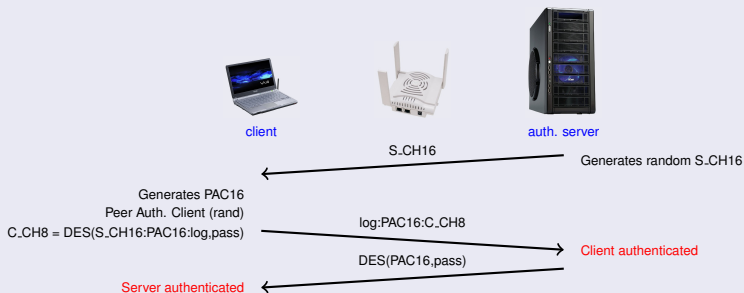
- 3-way handshake



# Authentication

## MSCHAP (*Micro\$oft Challenge Authentication Protocol*)

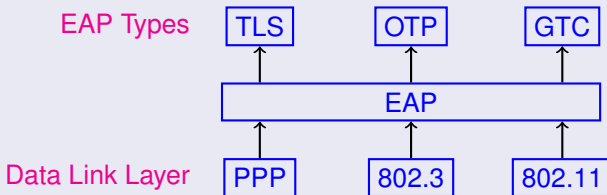
- v1. Same scheme than CHAP. DES cipher instead MD5
- v2. No backward compatibility. Authenticates server and client



# Authentication: EAP

## EAP (*Extensible Authentication Protocol*)

- Basis for 802.1x
- Initially thought for PPP (*Point to Point Protocol*)
- **Framework** supporting different authentication mechanisms: MD5, *One Time Password*, TLS, TTLS, LEAP, ...



From: Wireless Networks, The Definitive Guide, O'Reilly, 2002

# Authentication: EAP

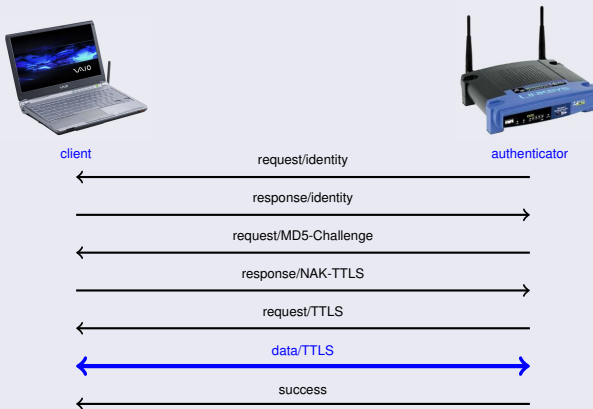
## EAP: Frame

- **Code:** request (1), response (2), success(3)
- **Identifier:** It is the same for the corresponding *requests* and *responses*
- **Length**
- **Type:**
  - 1 Identity
  - 2 Notification
  - 3 NAK, suggests a new authentication method
  - 4 MD5
  - 5 OTP
  - 6 GTC
  - 13 TLS
  - 21 TTLS
- **Data**



# Authentication: EAP

## EAP: example



# Authentication: EAP

## EAP: example

eap.cap - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: eap

No.	Time	Source	Destination	Protocol	Info	Length
1	0.000000	Cisco-Li_7a:e4:87		EAP	Request, Identity [RFC3748]	25
16	2.756013	Cisco-Li_7a:e4:87		EAP	Request, Identity [RFC3748]	25
17	2.756208	IntelCor_55:e5:4		EAP	Response, Identity [RFC3748]	30
18	2.781338	Cisco-Li_7a:e4:87		EAP	Request, MD5-Challenge [RFC3748]	42
19	2.781419	IntelCor_55:e5:4		EAP	Response, Legacy Nak (Response only) [RFC3748]	26
20	2.792906	Cisco-Li_7a:e4:87		EAP	Request, EAP-TTLS [RFC5281]	26
21	2.793339	IntelCor_55:e5:4		SSL	Client Hello	137
22	2.832718	Cisco-Li_7a:e4:87		EAP	Request, EAP-TTLS [RFC5281]	1044
23	2.832822	IntelCor_55:e5:4		EAP	Response, EAP-TTLS [RFC5281]	26
24	2.839506	Cisco-Li_7a:e4:87		TLSh1	Server Hello, Certificate, Server Key Exchange, Server Hello Do	657
25	2.880096	IntelCor_55:e5:4		TLSh1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Me	224
26	2.899002	Cisco-Li_7a:e4:87		TLSh1	Encrypted Handshake Message, Change Cipher Spec, Encrypted Hand	264
27	2.900457	IntelCor_55:e5:4		TLSh1	Application Data, Application Data	196
28	2.906984	Cisco-Li_7a:e4:87		TLSh1	Application Data	115
29	2.907189	IntelCor_55:e5:4		EAP	Response, EAP-TTLS [RFC5281]	26
30	2.916290	Cisco-Li_7a:e4:87		EAP	Success	24

Frame 16 (25 bytes on wire, 25 bytes captured)  
Linux cooked capture

802.1X Authentication

Version: 2  
Type: EAP Packet (0)  
Length: 5

Extensible Authentication Protocol

Code: Request (1)  
Id: 0  
Length: 5  
Type: Identity [RFC3748] (1)

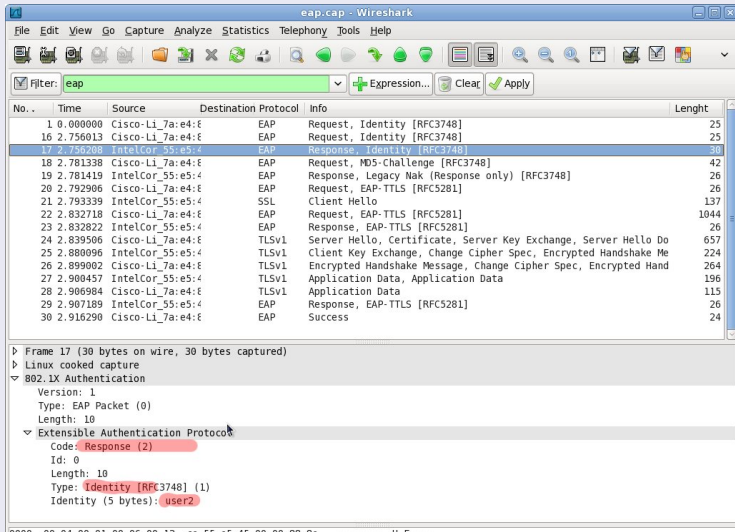
0000 00 00 00 01 00 06 00 13 10 7a e4 87 00 00 88 8e .....2.....  
0010 02 00 00 05 01 00 00 05 01





# Authentication: EAP

## EAP: example



The image shows a Wireshark capture of an EAP authentication process. The main packet list table is as follows:

No.	Time	Source	Destination	Protocol	Info	Length
1	0.000000	Cisco-Li_7a:e4:8		EAP	Request, Identity [RFC3748]	25
16	2.756013	Cisco-Li_7a:e4:8		EAP	Request, Identity [RFC3748]	25
17	2.756208	IntelCor_55:e5:4		EAP	Response, Identity [RFC3748]	30
18	2.781338	Cisco-Li_7a:e4:8		EAP	Request, MD5-Challenge [RFC3748]	42
19	2.781419	IntelCor_55:e5:4		EAP	Response, Legacy Nak (Response only) [RFC3748]	26
20	2.792906	Cisco-Li_7a:e4:8		EAP	Request, EAP-TTLS [RFC5281]	26
21	2.793339	IntelCor_55:e5:4		SSL	Client Hello	137
22	2.832718	Cisco-Li_7a:e4:8		EAP	Request, EAP-TTLS [RFC5281]	1044
23	2.832822	IntelCor_55:e5:4		EAP	Response, EAP-TTLS [RFC5281]	26
24	2.839506	Cisco-Li_7a:e4:8		TLSv1	Server Hello, Certificate, Server Key Exchange, Server Hello Do	657
25	2.880096	IntelCor_55:e5:4		TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Me	224
26	2.899002	Cisco-Li_7a:e4:8		TLSv1	Encrypted Handshake Message, Change Cipher Spec, Encrypted Hand	264
27	2.900457	IntelCor_55:e5:4		TLSv1	Application Data, Application Data	196
28	2.906984	Cisco-Li_7a:e4:8		TLSv1	Application Data	115
29	2.907189	IntelCor_55:e5:4		EAP	Response, EAP-TTLS [RFC5281]	26
30	2.916290	Cisco-Li_7a:e4:8		EAP	Success	24

Frame 17 (30 bytes on wire, 30 bytes captured)  
 Linux cooked capture  
 802.1X Authentication  
 Version: 1  
 Type: EAP Packet (0)  
 Length: 10  
 Extensible Authentication Protocol  
 Code: Response (2)  
 Id: 0  
 Length: 10  
 Type: Identity [RFC3748] (1)  
 Identity (5 bytes): user2



# Authentication: EAP

## EAP: example

Wireshark capture of an EAP authentication sequence. The packet list shows frames 1 through 30. Frame 18 is selected, showing an EAP Request, MD5-Challenge [RFC3748] (4) with a value of 2E02EAD5C11EBCC76E3D1486312F066B. The packet details pane shows the structure of the EAP packet, including the Extensible Authentication Protocol (EAP) and the MD5-Challenge.

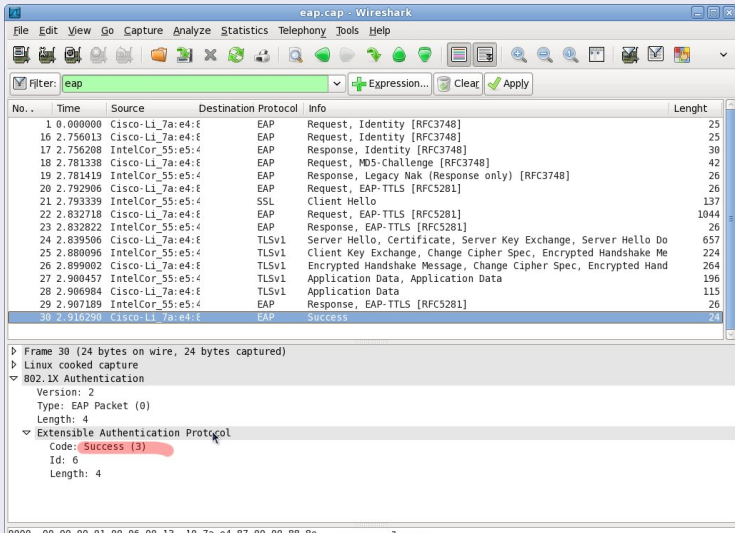
No.	Time	Source	Destination	Protocol	Info	Length
1	0.000000	Cisco-Li_7a:e4:8		EAP	Request, Identity [RFC3748]	25
16	2.756013	Cisco-Li_7a:e4:8		EAP	Request, Identity [RFC3748]	25
17	2.756208	IntelCor_55:e5:4		EAP	Response, Identity [RFC3748]	30
18	2.781338	Cisco-Li_7a:e4:8		EAP	Request, MD5-Challenge [RFC3748]	42
19	2.781419	IntelCor_55:e5:4		EAP	Response, Legacy Nak (Response only) [RFC3748]	26
20	2.792906	Cisco-Li_7a:e4:8		EAP	Request, EAP-TTLS [RFC5281]	26
21	2.793339	IntelCor_55:e5:4		SSL	Client Hello	137
22	2.832718	Cisco-Li_7a:e4:8		EAP	Request, EAP-TTLS [RFC5281]	1044
23	2.832822	IntelCor_55:e5:4		EAP	Response, EAP-TTLS [RFC5281]	26
24	2.839506	Cisco-Li_7a:e4:8		TLSv1	Server Hello, Certificate, Server Key Exchange, Server Hello Do	657
25	2.880096	IntelCor_55:e5:4		TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Me	224
26	2.899002	Cisco-Li_7a:e4:8		TLSv1	Encrypted Handshake Message, Change Cipher Spec, Encrypted Hand	264
27	2.900457	IntelCor_55:e5:4		TLSv1	Application Data	196
28	2.906984	Cisco-Li_7a:e4:8		TLSv1	Application Data	115
29	2.907189	IntelCor_55:e5:4		EAP	Response, EAP-TTLS [RFC5281]	26
30	2.916290	Cisco-Li_7a:e4:8		EAP	Success	24

Frame 18 (42 bytes on wire, 42 bytes captured)  
Linux cooked capture  
▼ 802.1X Authentication  
Version: 2  
Type: EAP Packet (0)  
Length: 22  
▼ Extensible Authentication Protocol  
Code: Request (1)  
Id: 1  
Length: 22  
Type: MD5-Challenge [RFC3748] (4)  
Value-Size: 16  
Value: 2E02EAD5C11EBCC76E3D1486312F066B



# Authentication: EAP

## EAP: example



The image shows a Wireshark capture of an EAP authentication process. The packet list shows 30 packets, with the last packet (No. 30) being an EAP Success message. The packet details pane shows the structure of the EAP Success message, including the Extensible Authentication Protocol (EAP) and the Success code.

No.	Time	Source	Destination	Protocol	Info	Length
1	0.000000	Cisco-Li_7a:e4:e		EAP	Request, Identity [RFC3748]	25
16	2.756013	Cisco-Li_7a:e4:e		EAP	Request, Identity [RFC3748]	25
17	2.756208	IntelCor_55:e5:4		EAP	Response, Identity [RFC3748]	30
18	2.781338	Cisco-Li_7a:e4:e		EAP	Request, MD5-Challenge [RFC3748]	42
19	2.781419	IntelCor_55:e5:4		EAP	Response, Legacy Nak (Response only) [RFC3748]	26
20	2.792906	Cisco-Li_7a:e4:e		EAP	Request, EAP-TTLS [RFC5281]	26
21	2.793339	IntelCor_55:e5:4		SSL	Client Hello	137
22	2.832718	Cisco-Li_7a:e4:e		EAP	Request, EAP-TTLS [RFC5281]	1044
23	2.832822	IntelCor_55:e5:4		EAP	Response, EAP-TTLS [RFC5281]	26
24	2.839506	Cisco-Li_7a:e4:e		TLSv1	Server Hello, Certificate, Server Key Exchange, Server Hello Do	657
25	2.880096	IntelCor_55:e5:4		TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Me	224
26	2.899002	Cisco-Li_7a:e4:e		TLSv1	Encrypted Handshake Message, Change Cipher Spec, Encrypted Hand	264
27	2.900457	IntelCor_55:e5:4		TLSv1	Application Data, Application Data	196
28	2.906984	Cisco-Li_7a:e4:e		TLSv1	Application Data	115
29	2.907189	IntelCor_55:e5:4		EAP	Response, EAP-TTLS [RFC5281]	26
30	2.916290	Cisco-Li_7a:e4:e		EAP	Success	24

Frame 30 (24 bytes on wire, 24 bytes captured)  
 Linux cooked capture  
 802.1X Authentication  
 Version: 2  
 Type: EAP Packet (0)  
 Length: 4  
 Extensible Authentication Protocol  
 Code: Success (3)  
 Id: 6  
 Length: 4



# Authentication: 802.1x

## 802.1x

- Defines a **port-based** authentication method
- Initially thought for 802.3 (*Ethernet*)
- Also known as EAPOL (*EAP over LAN*)
- Main parts:
  - **Supplicant**: Client
  - **Authenticator**: Can be the AP or the WiFi Controller
  - **AAA Server**: (*Accounting, Authorization and Authentication Server*). Can be a RADIUS server



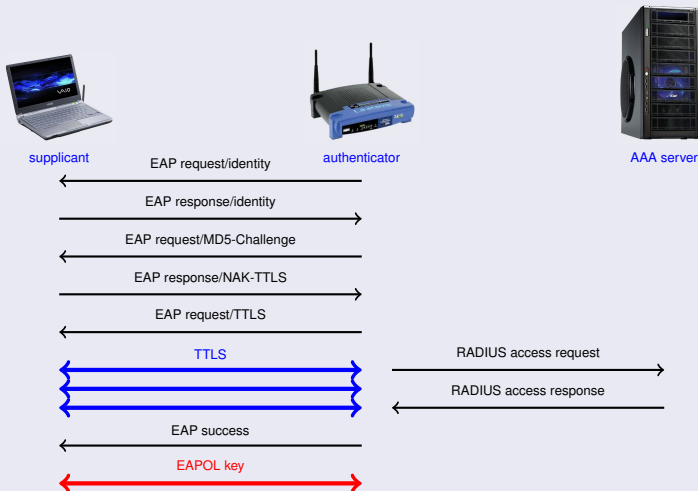
# Authentication: 802.1x

## Frame Format

- **MAC Header:** source and destination addresses
- **Type:** Ethernet
- **Version:** 1
- **Frame type:**
  - EAP-Packet (EAP Frames)
  - EAPOL-Start (*Supplicant* starts the process when no started by the *Authenticator*)
  - EAPOL-Logoff
  - EAPOL-Key (Crypto content for key establishment)
- **Length**
- **Frame body**

# Authentication: 802.1x

## Example





# Authentication: 802.1x

## EAP: example

radius.cap - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: radius

No.	Time	Source	Destination	Protocol	Info	Length
8	8.148455	192.168.202.1	192.168.202.2	RADIUS	Access-Request(1) (id=0, l=123)	165
9	8.158426	192.168.202.2	192.168.202.1	RADIUS	Access-challenge(11) (id=0, l=98)	140
10	8.154259	192.168.202.1	192.168.202.2	RADIUS	Access-Request(1) (id=0, l=137), Duplicate Request ID:0	179
11	8.156048	192.168.202.2	192.168.202.1	RADIUS	Access-challenge(11) (id=0, l=82)	124
12	8.161855	192.168.202.1	192.168.202.2	RADIUS	Access-Request(1) (id=0, l=240), Duplicate Request ID:0	250
13	8.189294	192.168.202.2	192.168.202.1	RADIUS	Access-challenge(11) (id=0, l=108)	1150
14	8.193974	192.168.202.1	192.168.202.2	RADIUS	Access-Request(1) (id=0, l=137), Duplicate Request ID:0	179
15	8.195744	192.168.202.2	192.168.202.1	RADIUS	Access-challenge(11) (id=0, l=717)	759
16	8.252326	192.168.202.1	192.168.202.2	RADIUS	Access-Request(1) (id=0, l=335), Duplicate Request ID:0	377
17	8.268327	192.168.202.2	192.168.202.1	RADIUS	Access-challenge(11) (id=0, l=320)	362
18	8.278554	192.168.202.1	192.168.202.2	RADIUS	Access-Request(1) (id=0, l=307), Duplicate Request ID:0	349
19	8.279638	192.168.202.2	192.168.202.1	RADIUS	Access-challenge(11) (id=0, l=187)	229
20	8.283676	192.168.202.1	192.168.202.2	RADIUS	Access-Request(1) (id=0, l=137), Duplicate Request ID:0	179
21	8.284378	192.168.202.2	192.168.202.1	RADIUS	Access-Accept(2) (id=0, l=185)	227

▶ AVP: l=14 t=NAS-Identifier(32): 001310/ae48/  
 ▶ AVP: l=6 t=NAS-Port(5): 53  
 ▶ AVP: l=6 t=Framed-MTU(12): 1400  
 ▶ AVP: l=18 t=State(24): 78F8AA4079FABF1660CB0EF94E75C554  
 ▶ AVP: l=6 t=NAS-Port-Type(61): Wireless-802.11(19)  
 ▶ AVP: l=119 t=EAP-Message(79) Last Segment[1]

EAP Fragment

- Extensible Authentication Protocol
  - Code: Response (2)
  - Id: 2
  - Length: 117
  - Type: EAP-TTLS [RFC5281] (21)
  - Flags(0x0):
  - TTLS version 0
- Secure Socket Layer
  - TLSv1 Record Layer: Handshake Protocol: Client Hello
    - Content Type: Handshake (22)
    - Version: TLS 1.0 (0x0301)
    - Length: 106
  - Handshake Protocol: Client Hello
    - Handshake Type: Client Hello (1)
    - Length: 102
    - Version: TLS 1.0 (0x0301)

0000 00 90 27 2c b3 97 00 13 10 7a ea 85 08 00 45 00 ...:.....E..  
 0010 01 14 09 96 40 08 40 11 1a ee c0 a8 ca 01 c0 a8 ...:.....





# WEP, Wireless Enhanced Privacy

## WEP facts

802.11 uses WEP as cipher algorithm

- **PRNG** (*Pseudo-Random Number Generator*)
- **Stream cipher** method **RC4** (*Rivest Cryptosystem number 4*)
- Broken in 2000

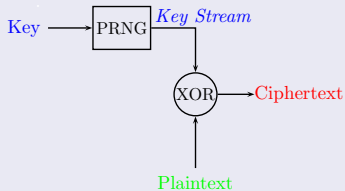
## WEP Keys

- 4 keys. Only one used for cipher/decipher
- Two ways of generation:
  - **Static**: The key is written on each device
  - **Dynamic**: Generated from a **pass phrase**

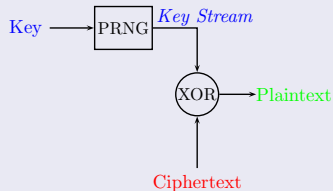
# WEP, Wireless Enhanced Privacy

## WEP Cipher method

Based on RC4



*Cipher process*

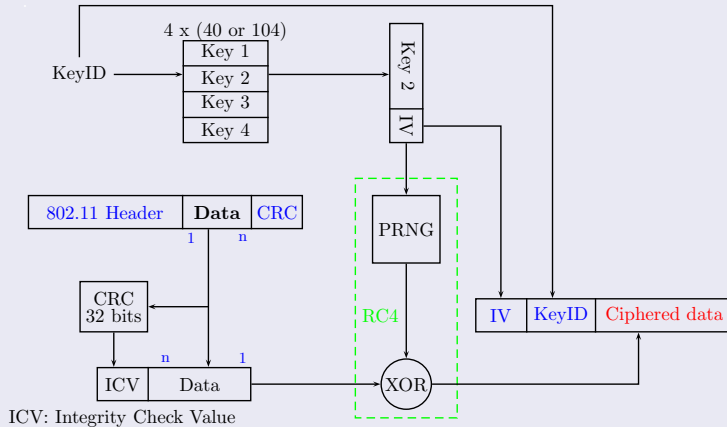


*Decipher process*

# WEP, Wireless Enhanced Privacy

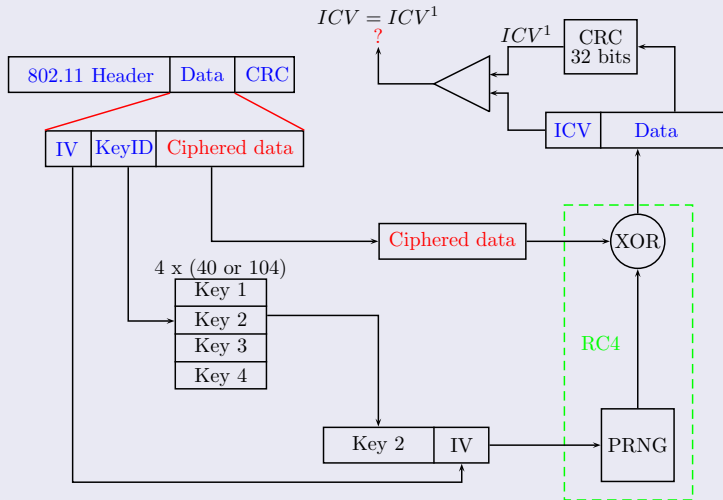
## WEP Cipher method

Based on RC4



# WEP, Wireless Enhanced Privacy

## WEP Decipher method



# 802.11i

## 802.11i facts

- Standard for wireless network security. Approved in July, 2004
- Improves security pitfalls on previous mechanisms
- WPA (*WiFi Protected Access*) is a subset of 802.11i
- Security methods
  - RSNA (Robust Security Network Association):
    - WPA: WEP + TKIP (Temporal Key Integrity Protocol)
    - WPA2: AES128 + CCMP (Counter Mode with CBC-MAC Protocol)
    - WPA3: AES128 + CCMP in personal mode. AES256-GCM-SHA384 (GCM: Galois Counter Mode) in enterprise mode. Replaces PSK by SAE (Simultaneous Authentication of Equals)
    - Key management
  - Pre-RSNA. (WEP)
- WPA2/3 modes may authenticate using PSK (Pre-Shared Keys) (WPA-Personal) or 802.1x (WPA-Enterprise)



# 802.11i

## TKIP (Temporal Key Integrity Protocol)

- Creates a MIC (Message Integrity Code) based on frame data and header to enable authentication (Michael algorithm)
- Uses the temporal key obtained from key management (see later) to derive RC4 key and WEP seed
- MIC prevent attacks based on:
  - bit flipping
  - iterative guessing of the key
  - redirection and impersonation (changing MAC addresses)



# 802.11i

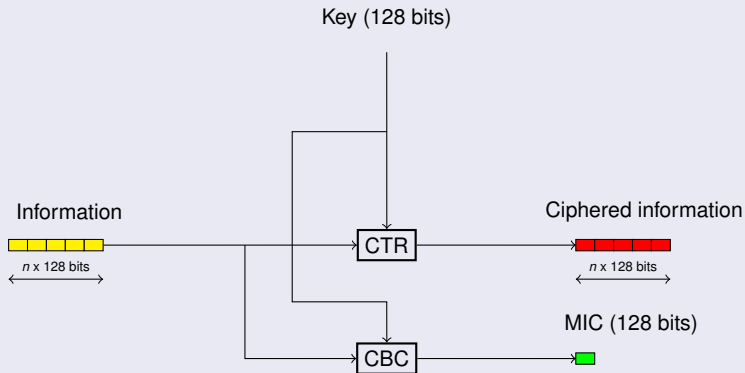
## Counter Mode with CBC-MAC Protocol (CCMP)

- AES (Advanced Encryption Standard) based cipher method. Cipher function with 128, 192 or 256 bits length security
- Algorithm CCM (Counter mode with CBC-MAC) employed, giving privacy, integrity and authentication
- CCMP is mandatory when 802.11i employed
- Two CCM **cipher modes**: CTR (*Counter*) and CBC (*Cipher Block Chaining*) using the **same key** with 2 different objectives:
  - 1 CTR. Ciphers information. Converts a block cipher as AES into a stream cipher mechanism
  - 2 CBC. Uses the last ciphered block as a Message Integrity Code (MIC)



## 802.11i

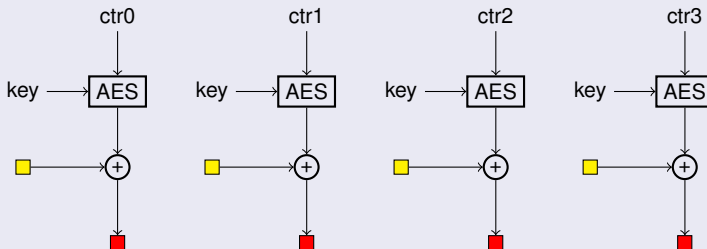
## CCM outline





# 802.11i

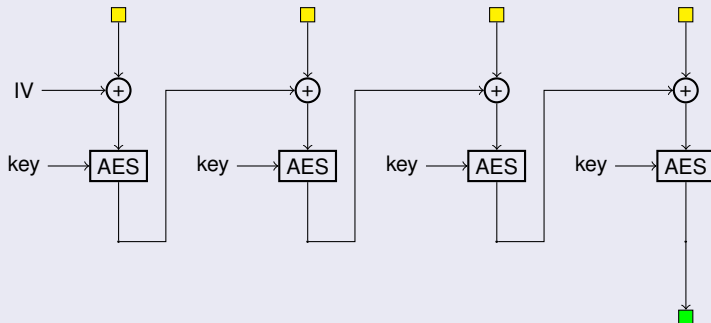
## CCM: CTR mode



- $ctr_i$ : 128 bits length block, no secret, different, generated for each key

# 802.11i

## CCM: CBC mode



- IV. Initialization vector.

0x00

# 802.11i

## Galois Counter Mode (GCM)

- Included in TLS1.3 and WPA3
- Authenticated encryption. Uses AAD (Additional and Authenticated Data) to generate a MIC (tag) field for authentication
- As in Counter Mode Ciphers (as CCMP) it is a stream cipher
- Operates in  $GF(2^n)$  where  $n$  is the block size (128,256, ...)
- AESGCM parameters for ciphering:
  - Key
  - Nonce (initializes  $IV$ )
  - AAD ( $A$ )
  - Plaintext

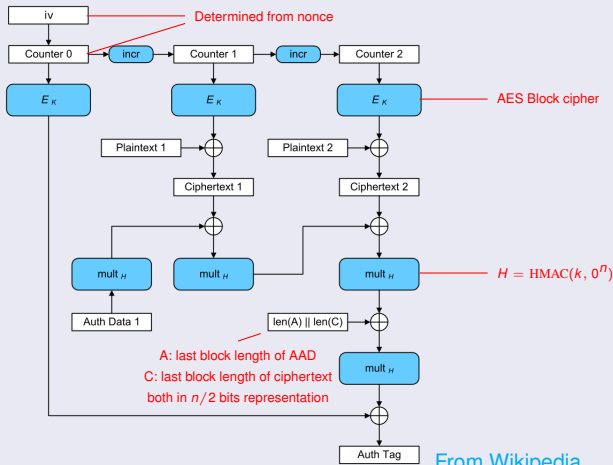
outputs:

- Ciphertext ( $C$ )
- Authentication tag



## 802.11i

## Galois Counter Mode (GCM)



From Wikipedia



# 802.11i

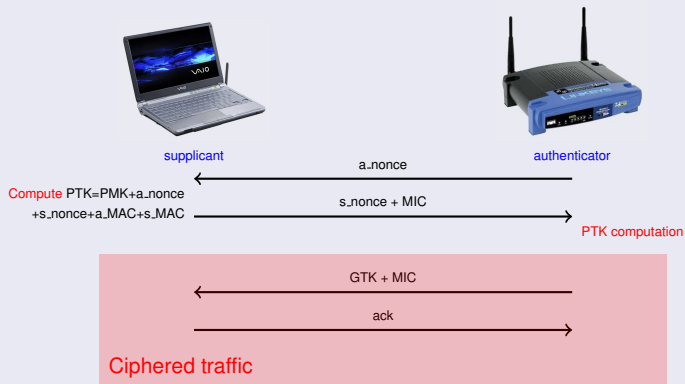
## Key management

- RSNA defines two key hierarchies:
  - Pairwise key. Protects unicast traffic
  - GTK (Group Temporal Key). Protects multicast and broadcast traffic
- Key exchange at frames **EAPOL-Key**
- First, **PMK (Pairwise Master Key)** is set at both ends based on:
  - Pre-shared keys (PSK) (WPA2-Personal)
  - 802.1x negotiation (WPA2-Enterprise)
- Then, **4-way handshake** is used to derive:
  - **PTK (Pairwise Temporal Key)**: protects handshake traffic
  - GTK
  - **TK (Temporal Key)**: protects unicast traffic
- Handshake may be periodically updated to renew keys in a session



## 802.11i

## 4-way handshake



PTK: Pairwise Temporal Key  
MIC: Message Integrity Code

GTK: Group Temporal Key



## 802.11i

## 4-way handshake example

Wireshark capture of an 802.11i 4-way handshake. The packet list shows an EAPOL Key packet (packet 18) from IntelCor\_55:e5:45 to Cisco-Li\_7a:e4:87. The packet details pane shows the Key Information and WPA Key fields.

No.	Time	Source	Destination	Protocol	Info	Length
15	0.138279	IntelCor_55:e5:45	Cisco-Li_7a:e4:87	EAP	Response, EAP-TTLS [RFC5281]	24
16	0.143751	Cisco-Li_7a:e4:87	IntelCor_55:e5:45	EAP	Success	22
17	0.144155	Cisco-Li_7a:e4:87	IntelCor_55:e5:45	EAPOL	Key	113
18	0.148600	IntelCor_55:e5:45	Cisco-Li_7a:e4:87	EAPOL	Key	135
19	0.152802	Cisco-Li_7a:e4:87	IntelCor_55:e5:45	EAPOL	Key	175
20	0.152903	IntelCor_55:e5:45	Cisco-Li_7a:e4:87	EAPOL	Key	113

Packet 18 details:

- Ethernet II, Src: Cisco-Li\_7a:e4:87 (00:13:10:7a:e4:87), Dst: IntelCor\_55:e5:45 (00:13:ce:55:e5:45)
- 802.1X Authentication
  - Version: 2
  - Type: Key (3)
  - Length: 117
  - Descriptor Type: EAPOL RSN key (2)
  - Key Information: 0x0089
    - ...001 = Key Descriptor Version: HMAC-MD5 for MIC and RC4 for encryption (1)
    - ...1... = Key Type: Pairwise Key
    - ...00... = Key Index: 0
    - ...0... = Install flag: Not set
    - ...1... = Key Ack flag: Set
    - ...0... = Key MIC flag: Not set
    - ...0... = Secure flag: Not set
    - ...0... = Error flag: Not set
    - ...0... = Request flag: Not set
    - ...0... = Encrypted Key Data flag: Not set
  - Key Length: 32
  - Replay Counter: 1
  - Nonce: CF9A8B043509023ED7A7875B841864B6108068305FF6FA
  - Key IV: 00000000000000000000000000000000
  - WPA Key RSC: 0000000000000000
  - WPA Key ID: 0000000000000000
  - WPA Key MIC: 00000000000000000000000000000000
  - WPA Key Length: 22
  - WPA Key: D014000FAC04A7DD03B081144932ACDC387B2B472278
    - Vendor Specific: Ieee8021: RSN
      - Tag Number: 221 (Vendor Specific)
      - Tag length: 20
      - Vendor: Ieee8021
      - Tag interpretation: RSN PMKID: A7DD03B081144932ACDC387B2B472278



## 802.11i

## 4-way handshake example

Wireshark capture of an EAPOL handshake (eapol.cap).

No.	Time	Source	Destination	Protocol	Info	Length
15	0.138279	IntelCor_55:e5:45	Cisco-L1_7a:e4:87	EAP	Response, EAP-TTLS [RFC5281]	24
16	0.143751	Cisco-L1_7a:e4:87	IntelCor_55:e5:45	EAP	Success	22
17	0.144175	Cisco-L1_7a:e4:87	IntelCor_55:e5:45	EAPOL	Key	135
18	0.148080	IntelCor_55:e5:45	Cisco-L1_7a:e4:87	EAPOL	Key	135
19	0.152802	Cisco-L1_7a:e4:87	IntelCor_55:e5:45	EAPOL	Key	175
20	0.152903	IntelCor_55:e5:45	Cisco-L1_7a:e4:87	EAPOL	Key	113

Descriptor Type: EAPOL RSN key (2)

Key Information: 0x0109

- .....001 = Key Descriptor Version: HMAC-MD5 for MIC and RC4 for encryption (1)
- .....1... = Key Type: Pairwise key
- .....00... = Key Index: 0
- .....0... = Install flag: Not set
- .....0... = Key Ack flag: Not set
- .....1... = Key MIC flag: Set
- .....0... = Secure flag: Not set
- .....0... = Error flag: Not set
- .....0... = Request flag: Not set
- .....0... = Encrypted Key Data flag: Not set

Key Length: 0

Replay Counter: 1

Nonce: 00043001A0426213836C756498EF91180A5DE4E8A7537C...

Key IV: 00000000000000000000000000000000

WPA Key RSC: 0000000000000000

WPA Key ID: 0000000000000000

WPA Key MIC: 2C2AC793CBF233EA9EDAFC970EB7C42

WPA Key Length: 22

WPA Key: 30140100000FAC020100000FAC020100000FAC010000

RSN Information

- Tag Number: 48 (RSN Information)
- Tag length: 20
- Tag interpretation: RSN IE, version 1
- Tag interpretation: Multicast cipher suite: TKIP
- Tag interpretation: # of unicast cipher suites: 1
- Tag interpretation: Unicast cipher suite 1: TKIP
- Tag interpretation: # of auth key management suites: 1
- Tag interpretation: auth key management suite 1: WPA

RSN Capabilities: 0x0000

0000 00 13 10 7a e4 87 00 13 ce 55 e5 45 88 8e 01 03 ...Z...U.E...

0010 00 75 02 01 09 00 00 00 00 00 00 00 00 01 3b ...U.....

0020 00 43 90 1a d4 28 21 28 3e 8c 75 b4 38 ef 91 18 ...C...I...>U...





## 802.11i

## 4-way handshake example

Wireshark capture of an 802.11i 4-way handshake. The packet list shows frames 15 to 20. Frame 19 (175 bytes) is selected, showing the 802.1X Authentication details. The details pane shows the EAPOL RSN key exchange, including the Key Descriptor, Key Information, and the encrypted key data.

No.	Time	Source	Destination	Protocol	Info	Length
15	0.138279	IntelCor_55:e5:45	Cisco-Li_7a:e4:87	EAP	Response, EAP-TTLS [RFC5281]	24
16	0.143751	Cisco-Li_7a:e4:87	IntelCor_55:e5:45	EAP	Success	22
17	0.144175	Cisco-Li_7a:e4:87	IntelCor_55:e5:45	EAPOL	Key	135
18	0.148600	IntelCor_55:e5:45	Cisco-Li_7a:e4:87	EAPOL	Key	135
19	0.152802	Cisco-Li_7a:e4:87	IntelCor_55:e5:45	EAPOL	Key	175
20	0.152903	IntelCor_55:e5:45	Cisco-Li_7a:e4:87	EAPOL	Key	113

Frame 19 (175 bytes on wire, 175 bytes captured)

Ethernet II, Src: Cisco-Li\_7a:e4:87 (00:13:10:7a:e4:87), Dst: IntelCor\_55:e5:45 (00:13:ce:55:e5:45)

802.1X Authentication

Version: 2

Type: Key (3)

Length: 157

Descriptor Type: EAPOL RSN key (2)

Key Information: 0x13c9

- 001 = Key Descriptor Version: HMAC-MD5 for MIC and RC4 for encryption (1)
- 1... = Key Type: Pairwise key
- 00... = Key Index: 0
- 1... = Install flag: Set
- 1... = Key Ack flag: Set
- 1... = Key MIC flag: Set
- 1... = Secure flag: Set
- 0... = Error flag: Not set
- 0... = Request flag: Not set
- 1... = encrypted Key Data flag: Set

Key Length: 32

Replay Counter: 2

Nonce: CF99A8B043509823ED7A78758B418648B1080B83D5FF6FA...

Key IV: B1080BB3D5FF6FADC85B5869BEB38

WPA Key RSC: 3800000000000000

WPA Key ID: 0000000000000000

WPA Key MIC: 69995EF36FD844EFA2FABC3C3742577

WPA Key Length: 62

WPA Key: 60F4B32EBF9574F0FC87F854627069F54760E066494480...

0000 00 13 ce 55 e5 45 00 13 10 7a e4 87 88 Be 02 03 ...U.E...2.....

0010 00 9d 02 13 c9 00 20 00 00 00 00 00 02 cf ..... CP # 7x11 A K

0020 99 a8 b0 43 50 98 23 ed 7a 78 75 88 41 86 4b b1



# 802.11i

## 802.1x authentication and master key establishment

- Different authentication mechanisms supported by EAP
- Some of them also allow to establish secure keys at both ends
- TLS (*Transport Layer Security*). More reliable mechanism
- [Find here](#) more on TLS
- Problem: Certificates required in order to authenticate clients
- TTLS (*Tunneled TLS*) relax previous requirement (no client certificate required). An AAA server may be used for client authentication
- Both mechanisms end with the establishment of a 384 random bits block (**Master Block**), only known by both communicating parts

## 802.11i

## WPA2-Personal in detail (AES128-CCMP)

- PSK=PBKDF2HMAC(key='passphrase', algo=SHA1, salt=SSID,length=32,iteration=4096)
- data='Pairwise key expansion' || 0x00 || MAC\_add1 || MAC\_add2 || nonce\_1 || nonce\_2
- MAC\_add1, MAC\_add2, nonce\_1, nonce\_2 correspond to MAC addresses 1 and 2 of MAC header and nonces of handshake 1 and 2. (Ordered: less first)
- PTK = SHA1\_HMAC(key=PSK, in=data, length=48)
 

```
def SHA1_HMAC(key, in, length):
    r=""
    for i in [0:⌈ length/20 ⌉]: #(20 is the size of SHA1)
        r+=HMAC(key, algo=SHA1, in=in || i (1 byte))
    return r[:length]
```
- KCK=PTK[0:16]. EAPOL-Key Confirmation Key. Used to compute MIC on WPA EAPOL Key message
- KEK=PTK[16:32]. EAPOL-Key Encryption Key (KEK). Used to encrypt GTK
- TK=PTK[32:48]. Temporal Key (TK). Used to encrypt/decrypt Unicast data packets
- MIC for handshake messages from 2 to 4 is computed as:
  - $i \in [2, 4]$
  - mess\_i is the 802.1x authentication part of handshake message i
  - data='mess' || str(i) || mess\_i[0:81] || (0x00)\*16 || mess\_i[81+16:] (MIC field masked with 0)
  - MIC=HMAC(key=KCK, algo=SHA1, in=data)
  - If MIC == mess\_i[81:81+16] message is authentic



## 802.11i

## WPA2-Personal in detail (AES128-CCMP)

- Unwrap GTK
  - data is WPA Key Data in handshake message 3 (GTK AES wrapped)
  - $GTK = AES\_UNWRAP(key=KEK, in=data)[30:46]$
  - Used to encrypt/decrypt multicast/broadcast data packets
- Decrypt a data packet. AESCCMP requires:
  - A nonce (to initialize IV)
  - AAD (Additional Authenticated Data) to compute the MIC (also called tag)
- If frame has QoS field:
  - $AAD = FrameControl\ (2\ bytes) \ ||\ Address\_1\ (6\ bytes) \ ||\ Address\_2 \ ||\ Address\_3 \ ||\ 0x0000 \ ||\ QoSControl[0:2]$
  - $nonce = QoSControl[0:1] \ ||\ Address\_2 \ ||\ CCMPpar[0:2] \ ||\ CCMPpar[4:8]$
- If QoS doesn't exist:
  - $AAD = FrameControl \ ||\ Address\_1 \ ||\ Address\_2 \ ||\ Address\_3 \ ||\ 0x0000$
  - $nonce = 0x00 \ ||\ Address\_2 \ ||\ CCMPpar[0:2] \ ||\ CCMPpar[4:8]$
- CCMPpar is a 8 bytes field in 802.11 data frame. Implements an incremental counter
- $AESCCM\_decrypt(key=TK, tag\_length=8, in=encrypted\_data, nonce=nonce, associated\_data=AAD)$  **Unicast packets**
- $AESCCM\_decrypt(key=GTK, tag\_length=8, in=encrypted\_data, nonce=nonce, associated\_data=AAD)$  **Multicast/Broadcast packets**



## 802.11i

## WPA2-Enterprise in detail (TLS-AES256-GCM-SHA384)

## Getting the Master Key

- Even ciphersuite is TLS-AES256-GCM-SHA384:
  - AES256-GCM-SHA384 for tunneling encryption
  - WPA2 will use AES128-CCMP in the 4-way-handshake
- pm: decrypted premaster from Client Key Exchange
- `def PRF(key,label,seed,algo,n): # Pseudo Random Function`  
`seed=label || seed`  
`A0=seed , B=""`  
`for i in [1:n+1]:`  
`Ai= HMAC(key=pm,in=Ai-1,algo=algo)`  
`for i in [0:n]:`  
`B = B || HMAC(key=pm,in=Ai+1,algo=algo)`  
`return B`
- `s = SHA384(ClientHello || ServerHello || ServerCert || ServerHelloDone || ClientKeyExchange)`
- `MasterKey = PRF(key=pm,label="extended master secret",seed=s, algo=SHA384, n=1)[0:48]`



# 802.11i

## WPA2-Enterprise in detail (TLS-AES256-GCM-SHA384)

### Decrypting inner authentication (EAP-PAP/CHAP/...)

- AES256-GCM-SHA384 requires:
  - 64 bytes for client and server keys
  - 8 bytes for client and server IV
  - No MAC keys needed
- `s = ServerHello.random || ClientHello.random`
- `KeyBlock = PRF(key=MasterKey, label="key expansion", seed=s, algo=SHA384, n=2)[0:72]`
- `client_write_key = KeyBlock[0:32]`
- `server_write_key = KeyBlock[32:64]`
- `client_write_iv = KeyBlock[64:68]`
- `server_write_iv = KeyBlock[68:72]`
- Assume data being the encrypted EAP message
- `tag = data[-16:]` Authentication tag
- GCM Nonce has two parts:
  - Explicit: goes in the data. First 8 bytes
  - Implicit: derived from IV
- `client_nonce = client_write_iv || data[0:8]`



# 802.11i

## WPA2-Enterprise in detail (TLS-AES256-GCM-SHA384)

### Decrypting inner authentication (EAP-PAP/CHAP/...)

- `seq_num = 0x000000000000000001`
- `content_type = 0x17`
- `version = 0x0303`
- `ciphertext_length = len(data)-24` (2 bytes). 8 explicit nonce + 16 tag
- `AAD = seq_num || content_type || version || ciphertext_length`
- `AESGCM.decrypt(in=data[8:],nonce= client_nonce , key=client_write_key, associated_data=AAD)`

# 802.11i

## WPA2-Enterprise in detail (TLS-AES256-GCM-SHA384)

### 4-way handshake

- $s = \text{ClientHello.random} \parallel \text{ServerHello.random}$
- $\text{KeyBlock} = \text{PRF}(\text{key}=\text{MasterKey}, \text{label}=\text{"tls keying material"}, \text{seed}=s, \text{algo}=\text{SHA384}, n=1)$
- $\text{MSK} = \text{KeyBlock}[0:48]$
- $\text{data} = \text{'Pairwise key expansion'} \parallel 0x00 \parallel \text{MAC\_add1} \parallel \text{MAC\_add2} \parallel \text{nonce\_1} \parallel \text{nonce\_2}$
- $\text{MAC\_add1}, \text{MAC\_add2}, \text{nonce\_1}, \text{nonce\_2}$  correspond to MAC addresses 1 and 2 of MAC header and nonces of handshake 1 and 2. (Ordered: less first)
- $\text{PTK} = \text{SHA1\_HMAC}(\text{key}=\text{MSK}[0:32], \text{in}=\text{data}, \text{length}=48)$
- KCK, KEK and TK as before
- Handshake messages authenticated as before
- GTK decrypted as before
- Data packet decrypted as before





# Contents

- 1 Wireless Ethernet
- 2 WiFi Security
- 3 Wifi deployments**
  - Centralized control architecture
  - Alcatel-Lucent Instant AP
- 4 Bibliography



# Wifi deployments

## The challenges

A medium/large size wifi deployment presents some challenges:

### 1 Coverage

- Limited coverage range per AP (less than 100 mts). Infrastructure/building dependent
- Trade-off between density deployment and cost
- Radio Frequency plan required to avoid interferences
- Network backbone determines AP situation. Mesh networks

### 2 Capacity

- Some spot areas may require higher bandwidth. Dual 802.11 b/g and 802.11a deployments
- Roaming may be required. IP mobility
- Non stationary scenario. People moves, so bandwidth requirement changes

### 3 Security

- Per user based security profiles (students, staff, visitors, ...)
- Don't rely on weak encryption methods
- Integration to the corporate authenticators



# Wifi deployments

## Two approaches

### • Fat APs

- Suitable for small deployments
- AP contains full configuration parameters
- Cheap solution but hard to manage
- Open source initiatives [DD-WRT](#)

### • Centralized control and Thin APs

- AP boots configuration from controller
- APs merely radio repeaters
- Controller have the full network configuration
- Many integrated features: firewall, automatic radio management, radio frequency planner, ...
- Easy to manage
- Proprietary solutions: Alcatel-Lucent, Aruba, Cisco, ...



# Centralized control architecture

## The components

### ● Controller

- Performs all the configuration and management tasks for WLAN
- Provides APs their config parameters
- Control over radio assignments (frequency and power), avoids interferences, central security checks
- Ends communication AP tunnels
- Must be sized according to the number of APs

### ● Thin APs

- Mere radio repeaters
- Boots from controller
- Multiple SSID (virtual AP)
- Consider 20/30 users per AP

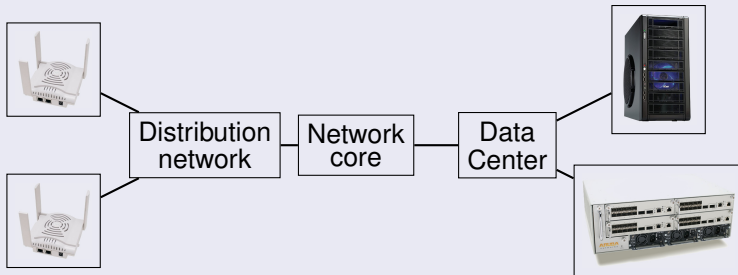
### ● **Software modules.** Define WLAN features: firewall, mesh, mobility, intrusion detection and response, adaptive radio management, ...

### ● **Licenses.** May be based on; number sessions, number APs, WLAN features, ...



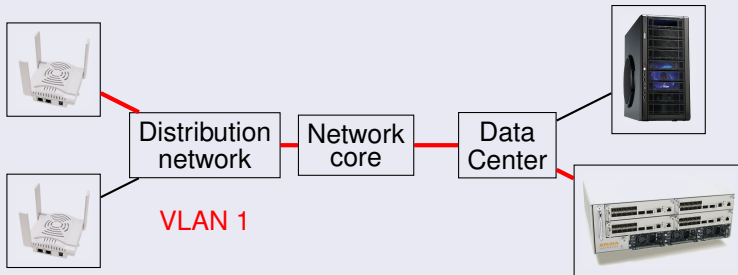
# Centralized control architecture

## Architecture outline



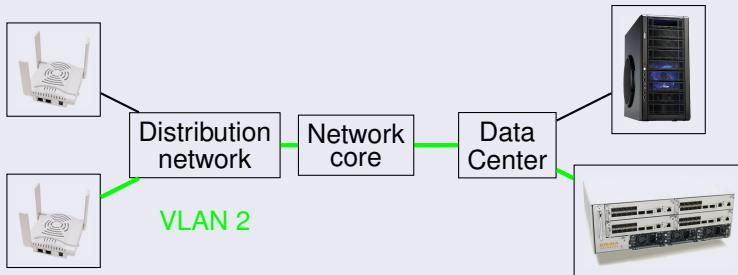
# Centralized control architecture

## Architecture outline



# Centralized control architecture

## Architecture outline



# Centralized control architecture

## AP booting process

- 1 AP must be able to obtain IP by DHCP
- 2 As well as the controller IP (statically or dynamically through DHCP option 41)
- 3 GRE tunnels are established between APs and controller
- 4 All AP traffic flows through controller (unless alternative AP operation modes specified; remote AP, ...)





# Alcatel-Lucent Instant AP

## Description

- Small deployments with centralized control not much expensive
- No controller needed. Control tasks assumed by an AP
- Up to 16 APs and 256 users per deployment
- APs family supported (IAP-105, 92, 93, 134 and 135)



# Alcatel-Lucent Instant AP

## Main features

- Easy configuration. Automatic master controller AP establishment (virtual controller)
- Web control interface
- Mesh networks support
- Authentication methods:
  - 802.1x
  - Captive portal
  - Based on MAC
- Firewall capabilities. Rules can be created
- Adaptive Radio Management (ARM)
  - Voice and load aware
  - Band steering (2.4 - 5 GHz)
  - Load balancing and air-time fairness
- Intrusion and Detection System (IDS). Rogue detection and containment methods



# Contents

- 1 Wireless Ethernet
- 2 WiFi Security
- 3 Wifi deployments
- 4 Bibliography**



# Bibliography

- 802.11 Wireless Networks: The Definitive Guide. Matthew Gast. Ed. O'Reilly, 2002.
- Implementing 802.1x. Security Solutions for Wired and Wireless Networks. Jim Geier. Wiley Publishing, 2008.
- 802.11i. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. 2016.
- [RFC 5281. Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 \(EAP-TTLSv0\)](#)

