Bonus Web Chapter 1

Introduction to 802.11 (Extended Web Edition)

The 802.11 standard defines a link-layer wireless protocol and is managed by the Institute of Electrical and Electronics Engineers (IEEE). Many people think of Wi-Fi when they hear 802.11, but they are not quite the same thing. In recent years, Wi-Fi and 802.11 have exploded in popularity, and every new laptop comes with a built-in Wi-Fi adapter. This popularity has led to a surge of research into the security of the 802.11 standard, which is covered throughout the book.

This Bonus Web Chapter lays the groundwork for a strong understanding of the 802.11 protocol. If you have some background with 802.11 and are interested in specific ways to attack or defend your network, you probably don't need this extended introduction. If you have never seen Wireshark display an 802.11 packet, or you are interested in some of the interesting features in the 802.11 Media Access Control (MAC) layer, read on.

802.11 History

The first 802.11 standard was approved in 1997 and allowed transmission speeds that topped out at 2 Mbps. This version of the standard allowed two different radio frequency encoding methods at a physical level: *Frequency Hopping Spread Spectrum (FHSS)* and *Direct Sequence Spread Spectrum (DSSS)*. These two different encoding schemes are incompatible, however, and the choice led to a lot of confusion in the marketplace.

In 1999, the IEEE released 802.11b, an amendment to the original 802.11 standard. The 802.11b standard used DSSS and increased the maximum transmission speed to a much faster 11 Mbps. Also released in 1999 was 802.11a, which allowed 802.11 to run outside of the crowded 2.4-GHz industrial, scientific, and medical (ISM) band and in the 5-GHz Unlicensed National Information Infrastructure (UNII) band. Due to increased cost and reduced signal propagation, 802.11a was not initially popular with manufacturers and consumers despite the increased speed it offered (54 Mbps).

Increasing the speed of 802.11 has been a consistent priority for the 802.11 committee, so in 2003, they released another speed boost, 802.11g, which brought 54 Mbps while also utilizing the 2.4-GHz band. The next speed increase, 802.11n, brings the theoretical throughput up to 300 Mbps (with many systems reaching 130–160 Mbps out-of-the-box) and was ratified in September 2009.

What About That Alleged IR Band for 802.11?

Some people wonder if you can use the IR port built in to many laptops to talk 802.11. In fact, the standard did include support for an IR physical layer (PHY), so theoretically, it is possible. However, very few products were produced that implemented the IR physical layer. The infrared ports on laptops are compatible with protocols designed by the Infrared Data Association (IrDA).

Wi-Fi vs. 802.11

Wi-Fi is a subset of the 802.11 standard that is managed by the Wi-Fi Alliance. Because the 802.11 standard is so complex, and the process required to update the standard can take awhile (it's run by a committee), nearly all of the major wireless equipment manufacturers decided they needed a smaller, more nimble group dedicated to maintaining interoperability among vendors while promoting the technology through marketing efforts. This resulted in the creation of the Wi-Fi Alliance.

The Wi-Fi Alliance assures that all products with a Wi-Fi-certified logo work together for a given set of functions. This way, if any ambiguity in the 802.11 standard crops up, the Wi-Fi Alliance defines the "right thing" to do. The alliance also allows vendors to implement important subsets of *draft standards* (standards that have not yet been ratified). The most well-known example of this is Wi-Fi Protected Access (WPA) or "draft" 802.11n equipment.

802.11 in a Nutshell

Most people know that 802.11 provides wireless access to wired networks with the use of an *access point (AP)*. In what is commonly referred to as *ad-hoc* or *Independent Basic Service Set (IBSS) mode*, 802.11 can also be used without an AP. Because those concerned about wireless security are not usually talking about ad-hoc networks, and because the details of the 802.11 protocol change dramatically when in ad-hoc mode, this section covers running 802.11 in infrastructure mode (with an AP), unless otherwise specified.

The 802.11 MAC

Some of the most interesting aspects of the 802.11 standard to security analysts are the rules defined for Media Access Control (MAC). Regardless of the physical layer (PHY) that 802.11 is implemented on (2.4-GHz ISM band, 5-GHz UNII band, and so on), the MAC rules stay the same.

Distributed Coordination Function

The 802.11 standard specifies two modes in which MAC can operate: *contention free* and *contention based*. In contention-based MAC, all stations compete for access to the media. Similar to Ethernet, when a station wants to transmit, it checks to see if another station is using the medium. In an Ethernet network, a station waits until the medium is not in use and then transmits the packet. If another station transmits at the same time, it will detect the collision and back off from transmitting by waiting for a randomly selected delay period. The ability to identify when the medium is in use and to detect a collision is what makes Ethernet a carrier-sense multiple-access/collision-*detection* (CSMA/CD) algorithm.

When 802.11 is operating in contention-based mode, it uses a similar technique. The biggest difference is that most 802.11 cards have only one radio, which means they can transmit or receive, but not do both at the same time, making collision detection impossible. Instead, 802.11 needs to employ collision *avoidance*, making the protocol CSMA/CA-based, not CSMA/CD. This mode, known as the *Distributed Coordination Function (DCF)*,

is the mode that almost all 802.11 networks operate under. In DCF mode, the station waits until the media is clear and then transmits data. After completing the transmission, the station waits for an acknowledgment message from the recipient to indicate the data was received successfully. If the acknowledgment message is not received, the data is retransmitted and marked to let the recipient know the station is sending the data again.

DCF and Multiple Recipients

Although the DCF mechanism to transmit and wait for an acknowledgment is effective at ensuring a single destination host has received the transmitted data successfully, the mechanism fails to accommodate traffic with multiple recipients. If a station is sending data as a broadcast packet (the destination MAC address is ff:ff:ff:ff:ff:ff) or a multicast packet (the data is sent to a group of recipients), it doesn't know how many hosts are on the network in order to identify if one or more stations missed the data. Further, if all the destination hosts were to return a positive acknowledgment simultaneously for a single received packet, it would cause massive collisions and pandemonium on the network. Appropriately, the IEEE 802.11 MAC specification indicates that positive acknowledgment messages should be sent only in response to *unicast data*, or traffic sent to a single destination host. This leaves stations transmitting broadcast or multicast traffic without an 802.11 mechanism to detect if their traffic was sent properly, instead relying on upper-layer protocols to determine if the station needs to retransmit, which is a performance detriment.

Point Coordination Function

Another mode in which the 802.11 MAC can operate is called the *Point Coordination Function (PCF)*. In this mode, the access point controls all access to the media. In some sense, this mode of operation is superficially similar to that of token ring; instead of stations passing around a token, however, the AP polls them to see if they have any data to transmit.

The greatest similarity between running an 802.11 network using the PCF or token ring is that of market share. We are unaware of any products that actually implement the PCF mode, but it persists in the standard. In the future, as 802.11 networks get more congested and collisions take up a significant amount of bandwidth, the PCF may be more widely deployed. Since no real-world networks use the PCF, the details of this medium management technique are largely omitted from this chapter.

Hybrid Coordination Function

The *Hybrid Coordination Function (HCF)* is used to implement QoS features in 802.11. When enabled, QoS allows users to send higher priority traffic (i.e., VoIP) with less latency than other traffic, such as an FTP transfer.

QoS implements a traffic prioritization management technique by allowing for a shorter contention window when high priority traffic is queued for transmission. For example, if a client wants to transmit FTP data, the client will wait for the channel to go quiet, choose a

random number between 0 and 40 (for example) slots, and wait this period of time. If the channel is still quiet after waiting for the randomly selected delay interval, the client will then transmit. If another client had VoIP data, this client would choose a random number between 0 and 20, allowing the VoIP traffic to take priority, but still allowing the FTP client access. QoS also enables improved performance via the use of Block ACKs and other more subtle techniques. These are implemented in the Hybrid Coordination Function.

Features of the 802.11 MAC

The 802.11 MAC is very complicated. There are two reasons for this. First, the standard is very ambitious. The type of MAC that is well suited to embedded systems is not necessarily well suited to laptops. The 802.11 MAC tries to be everything to everyone, and it appears to be succeeding—at least in terms of market share. Second, it has problems that have no wired-side analogy. The biggest of these are noisy links due to interference and hidden nodes. All of these reasons provide motivation for developing a link-layer standard brimming with problem-solving features.

Unfortunately, this surplus of features makes the standard a huge burden to implement correctly, which has led to many implementation bugs that have resulted in remote code execution vulnerabilities. This avalanche of features is only going to continue, however, as the IEEE continues to add them to an already complex protocol. Examples of complex 802.11 MAC enhancements include 802.11s Wireless Mesh Networking and 802.11v Wireless Network Management. Not all additional features being considered for 802.11 are detrimental to security; IEEE 802.11w is adding support for authenticating a subset of management frames.

Now that you understand the motivation for all of the 802.11 features, let's look at the core features that are universally implemented. This section focuses on the basics of the 802.11 protocol, as it can be found in the wild. For the sake of brevity, it largely ignores 802.11e QoS. If 802.11e is in use, things get significantly more complex.

Positive Acknowledgment

Before QoS was added to 802.11, all unicast traffic was positively acknowledged. Usually, positive acknowledgment is found in transport (layer four) protocols, such as TCP. Though it is true that reliable higher-layer protocols, such as TCP, would eventually cause a dropped 802.11 packet to be retransmitted, the 802.11 committee decided this would cause too much delay. With the addition of 802.11, clients maintain tighter control of acknowledgments.

A big advantage to having positive acknowledgment at the link layer is that it can be combined with fragmentation (or simply a small maximum transmission unit to begin with) to ensure only small amounts of data need to be retransmitted in case of a collision. Radio interference is quite often in small bursts. If these bursts occur during transmission of one small fragment instead of a large packet, less time is wasted on retransmission.

Fragmentation

One interesting 802.11 feature is that it is a link-layer protocol with support for fragmentation. Most network layer protocols (including IP) have support for fragmentation. When fragmenting at the network layer, however, the final destination must perform reassembly. Fragmenting at the link layer forces the next hop to perform reassembly.

Using fragmentation can help increase throughput across a noisy link. Instead of having to retransmit a single large frame when there is a collision or noise, the sender can break up the frame into many smaller fragments and only retransmit the fragments that get corrupted. Of course, having a MTU set too small, or sending unnecessarily small fragments, negatively impacts throughput due to the overhead associated with fragmentation and reassembly.

Aggregation

Not content with the ability to fragment packets, the 802.11 committee later decided glomming them together into mega-packets instead would be better. This feature was added in the 802.11n amendment. There are two different aggregation techniques: Aggregate-MAC Service Data Unit (A-MSDU) and Aggregate-MAC Protocol Data Unit (A-MPDU).

A-MSDU combines multiple packets to the same destination with a single header. A-MSDU is performed in hardware and can result in a single packet up to 8Kb in size. If there are any receiving errors, the entire aggregate packet will need to be retransmitted. A-MPDU is performed in software and transmits multiple packets to a single destination in one burst. A single A-MPDU can be up to 64KB, and individual elements can be acknowledged with a block acknowledgment. A-MPDU aggregation is performed in software.

Power Savings

Here's something you won't find in very many link-layer protocols: built-in support for power savings. Because the designers knew that most clients on a wireless network would be running on batteries, they included features to improve battery life.

Power savings in 802.11 works by letting clients turn their radios off during periods of inactivity. Basically, the clients inform the AP that they are disabling their radio, at which point the AP will buffer frames for the client. Sleeping stations must wake up periodically and examine beacon frames. These beacon frames carry a *Traffic Indication Map (TIM)*. A TIM is a bitmap that indicates which stations have buffered packets.

Once a station realizes that a packet is waiting for it, it transmits a PS-Poll frame. When the AP receives a PS-Poll, it transmits a single buffered frame back to the station. This process is repeated until all buffered frames have been received.

This protocol has a clever feature. Since the AP periodically transmits information about a client's buffered traffic, sleeping clients don't need to transmit any packets to discover they have packets waiting. A sleeping client can power up the receiver, discover it has no packets waiting, and power it back down, all without transmitting anything.

Multiple-Input Multiple-Output (MIMO)

MIMO is the feature that confuses most 802.11 users because it contradicts a basic notion they have been hearing for all these years. Only one transmitter can use the same channel at the same time. MIMO turns this on its head by utilizing multiple antennas for transmission and reception, on the same channel and at the same time. The number of independent transmit and receive chains is often written as 2x2 or 3x3. The first number is referring to the number of transmit chains, the second refers to receive chains. 802.11n accomplishes MIMO with two advanced signal-processing techniques: spatial multiplexing and space-time block coding.

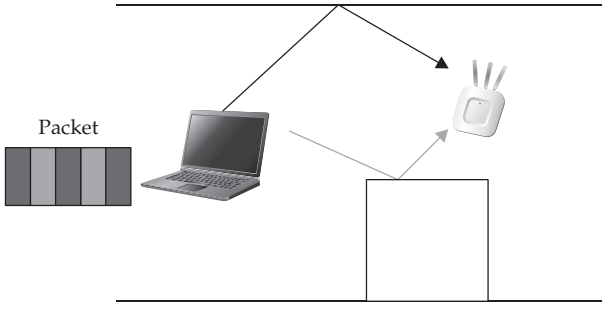
Spatial multiplexing requires both the transmitter and receiver to each have at least two antennas. Spatial multiplexing works by dividing a single packet into two different bit streams. These streams are transmitted out both antennas at the same time. Each antenna is said to transmit its own spatial stream. The hope is that each antenna will be oriented differently enough that spatial streams from each antenna will arrive at different times and with different strengths. The receiving end will be able to distinguish the streams according to the various delays and strengths and put the bits back together. When people refer to 802.11n taking advantage of multipath, this is the feature they are talking about.

Imagine you were trying to transmit this paragraph to someone by saying it out loud. In a 2x2 MIMO system, you would need to split the paragraph up, say by every other word, and then say both sets of words out loud and simultaneously (possibly with the help of a friend). This is a trick that humans find very hard to do, but the engineers behind 802.11n have figured out how to make radios do it.

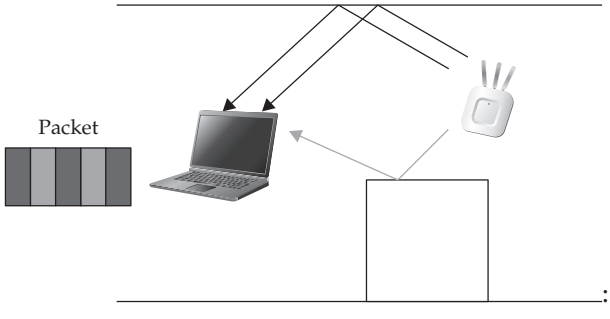
Space-time block coding can be used when the transmitter has more antennas available than the receiver. Most 802.11n APs currently have three antennas (four are possible), whereas most 802.11n clients only have two antennas. Rather then let one antenna sit idle, the third antenna can be used to transmit a redundant copy of a spatial stream, allowing the receiver to more easily handle any errors in transmission. Space-time block coding is optional.

Both of these features are shown in the following illustration.

MIMO: Spacial multiplexing



MIMO: Space-time block coding



40-MHz Channels

Before 802.11n was created, all APs were tuned to a single 20-MHz or 22-MHz wide channel. Although users can select from channels 1–11 in the 2.4-GHz ISM band, really only three (1, 6, and 11) don't overlap. The 5-GHz UNII band is less crowded, allowing for 19 nonoverlapping 20-MHz channels.

By doubling the beamwidth used, the 802.11n designers could easily double the throughput. The only downside is that in the 2.4-GHz range, there is only enough room for one nonoverlapping 40-MHz channel plus one 20-MHz channel. This means that any 40-MHz network operating at 2.4 GHz is almost definitely going to step on neighbors. This requires a complicated set of features to use only 40-MHz channels in order to not pulverize other networks. A 40-MHz network can operate in three modes: Greenfield (no concern for legacy b/g clients), mixed mode (transmit a header at b/g rates that keep the media reserved), and legacy mode (utilize a CTS to self).

Due to the lack of bandwidth in the 2.4-GHz band, not many 40-MHz networks will be found. (Support for 40-MHz channels is optional in the standard.) Nonetheless, if you are interested in capturing traffic on 802.11n networks, you will probably want a card that supports a 40-MHz operation mode.

Things in the 5-GHz UNII band are not as crowded as in the 2.4-GHz band. There is enough room for 19 nonoverlapping 20-MHz channels here, which means there is room enough for nine 40-MHz channels. In the future, 40-MHz networks in the 5-GHz band will probably become more popular.

RTS/CTS Packets and the Hidden Node Problem

One of the unique aspects of 802.11 is that two nodes can be connected to the same AP, but not hear each other's transmissions. This is called the *hidden node problem*. Imagine a single AP in the middle of a large hanger, with two nodes (A and B) on either end. The nodes may not be able to hear each other's radios, but the AP can. If these two clients attempt to transmit at the same time (because the channel appears to be idle), they will collide at the AP in the middle.

To avoid collisions, the 802.11 includes two interesting control packets: *Request to Send (RTS)* and *Clear to Send (CTS)*. If client A wants to avoid a collision like the one just outlined, he transmits a RTS to the AP. The AP will then respond with a CTS. The CTS packet tells everyone in range (except station A) not to transmit for a specified duration in microseconds. Because station B can hear the CTS coming from the AP, station B won't transmit during station A's timeslot, and the collision is avoided.

802.11 Packet Types

The 802.11 standard divides all packets into three different categories: data, management, and control. These different categories are known as the *type of packet.* There are many different subtypes for a given packet type. Beacons and deauthentication packets are both examples of management packet subtypes. In order to understand why many of the attacks covered in the book work, knowing the differences among the packet types is helpful.

Control Packets

Control packets are the lowest level of packet type. They are called control packets because they are directly related to the standard's Media Access Control (MAC) rules. Currently, the standard defines six different control packets, shown in Table 1.

Two of these control frames are directly related to the PCF mode of operation mentioned previously (CF-End and CF-End + CF-Ack). For all practical purposes, these frames are currently unused.

As mentioned in the previous section, RTS/CTS packets help solve the hidden node problem. They can also be used to avoid collisions even when the node is not hidden. When a station wants to transmit a large packet, even without hidden nodes, a collision is possible. Instead of transmitting a large packet, the station can send an RTS. If the (relatively small) RTS packet gets lost in a collision, little time is wasted retransmitting it.

Once the station receives the AP-generated CTS, it can transmit the large packet without worrying about a collision.

Block Acknowledgment (BA) and Block Acknowledgment Request (BAR) packets were introduced as an optional part of the 802.11e QoS specification and later as a mandatory component of 802.11n high-throughput support. Instead of sending an ACK for each frame received, a client can send a BA with a bitmap indicating successful or failed delivery for multiple frames. The BA can be sent unsolicited by a client or in response to a BAR.

There are only two control packets left: PS-Poll and acknowledgments. As mentioned previously, PS-Poll packets are used by clients to retrieve buffered packets from the AP when the client is in power-savings mode. There isn't much to say about acknowledgments. Acknowledgment packets are small, and they are used to acknowledge the receipt of unicast data and some management packets.

The most interesting thing about control packets is that some of them are explicitly designed to be honored by unrelated networks on the same channel. This means that if you and your neighbor have your own networks, and your AP sends out a Clear To Send (CTS), all of the 802.11 nodes that hear it (including your neighbors) are expected to honor the CTS packet and not transmit anything for the duration specified.

Type	Subtype	Description
Control	8	Block Ack Request (QoS)
Control	9	Block Ack (QoS)
Control	10	Power Save (PS)-Poll
Control	11	Request to Send (RTS)
Control	12	Clear to Send (CTS)
Control	13	Acknowledgment (ACK)
Control	14	Contention-Free (CF)-End
Control	15	CF-End + CF-Ack

Table 1 Currently Defined Control Packets

The fact that nodes on entirely unrelated networks are expected to process and honor certain packets from each other is interesting. It means that a small subset of the 802.11 protocol cannot, by design, be authenticated. Other examples of this include certain 802.11 Action frames (more on these later). This difference is a subtle but important one between 802.11 and virtually any other protocol on the planet.

If you hook up a computer to the Internet without a firewall (or a NAT), you have, in a sense, given anyone else on the Internet the ability to engage your computer in the TCP/IP protocol. If you think this is a bad idea (and obviously you should), you can turn on your own firewall, stick yourself behind a NAT, and so on.

Similarly, when you plug an Ethernet cable into your computer, you are giving everyone on the same broadcast domain the ability to engage your computer in the (relatively simple) layer two protocol, Ethernet. People generally don't worry about this for two reasons. One is that Ethernet is very simple, and therefore nobody has ever found a remotely exploitable bug in an Ethernet device driver. The other reason is that by virtue of being physically connected to the same wired network, there is some implied level of trust. Neither of these assurances applies to 802.11.

Management Packets

Management packets, like control packets, are also unauthenticated. However, because most management packets are only processed by stations on the same network, they could be authenticated in the future.

Management packets are used to perform various overhead tasks associated with running a wireless network, including such things as associating to a network and finding a network to associate with. Management frames that can generally be seen in the wild are shown in Table 2. Most of the packets in this table are covered in detail in "Finding and Connecting to Wireless Networks," later in this chapter.

Type	Subtype	Description
Management	0	Association request
Management	1	Association response
Management	2	Reassociation request
Management	3	Reassociation response
Management	4	Probe request
Management	5	Probe response
Management	8	Beacon
Management	10	Disassociation
Management	11	Authentication
Management	12	Deauthentication
Management	13	Action

Table 2 Important Management Packets

Data Packets

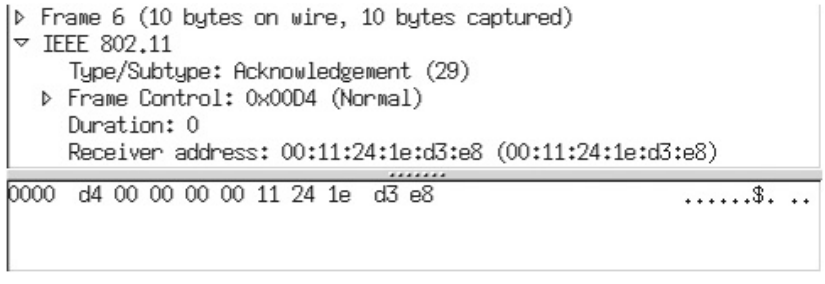
Data packets can be authenticated in 802.11, as long as some form of encryption is turned on. The strength of this authentication is strictly related to the strength of the encryption being used. WEP provides very little assurance that the packet actually originated from someone on your network. WPA provides a much stronger guarantee.

Before QoS was introduced, there were eight different subtypes for data packets. Almost all of these are due to the (currently unused) PCF mode of operation. Practically speaking, data packets on a non-QoS network have only two subtypes: Subtype 0 indicates a normal data packet, and subtype 4 indicates a null function data packet. Null function data packets are most often used when a client has no data to transfer, but wants to inform the AP that it is changing its power-savings mode.

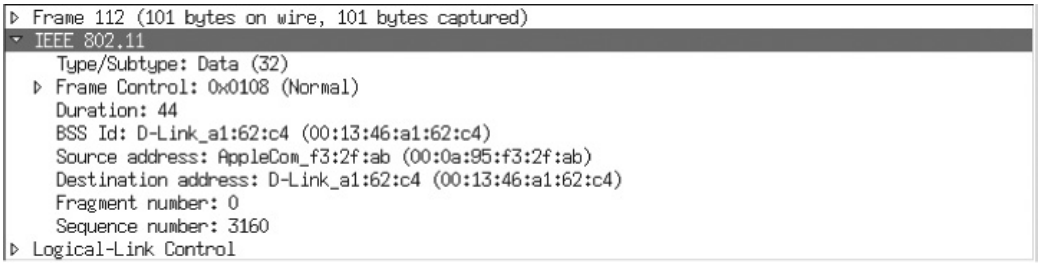
Addressing in 802.11 Packets

Unlike Ethernet, most 802.11 packets have three addresses: a source address, a destination address, and a *Basic Service Set ID (BSSID)*. The BSSID field uniquely identifies the AP and its collection of associated stations, and is often the same MAC address as the wireless interface on the AP. The three addresses tell the packets where they are going, who they came from, and what AP to go through.

Not all packets, however, have three addresses. Because minimizing the overhead of sending control frames (such as acknowledgments) is so important, the number of bits used is kept to a minimum. The IEEE also uses different terms to describe the addresses in control frames. Instead of a destination address, control frames have a receiver address, and instead of a source address, they have a transmitter address. The most common control frame is an Acknowledgment (ACK). The next illustration shows the Wireshark decoding of an ACK packet. Notice that it has only a single address, the receiver address. This is because an ACK packet, by definition, acknowledges the last packet sent. Unlike TCP, there is no need to identify exactly what is being acknowledged.



The next illustration shows a typical data packet. In this packet, the BSSID and destination address are the same because the packet was headed to an upstream network, and the AP was the default gateway. If the packet had been destined for another machine on the same wireless network, the destination address would be different than the BSSID.



Interesting Fields Across Packets

All 802.11 packets have a certain set of fields, regardless of whether they are data frames or control/management frames. This section covers the fields that are carried across all 802.11 packets.

Version All packets carry a 2-bit Version field. Currently, the only defined value is 0.

Type/Subtype These two combined fields uniquely determine what sort of packet you are looking at. For example, `type = 0, subtype = 8` indicates that this is a management packet (`type 0`) that is a beacon (`subtype 8`). The type value of 3 ("11" in binary) is not used. Many subtype values are also reserved for the various frame types.

ToDS/FromDS These two bits indicate whether a packet is coming from or going to the Distribution System (DS), logically the AP or the backend network infrastructure. These bits are only relevant to data packets; all management and control-type packets are supposed to set these bits to 0. If both bits are 1, then the packet is actually a wireless distribution system (WDS) packet being forwarded from one AP to another. If both bits are 0 and the type is data, then the packet is from an ad-hoc network. When only the FromDS field is set, then it is a packet from the AP to a client. If only ToDS is set, the packet is from the client to the AP. When performing traffic injection attacks, it is important to know if the traffic should be injected toward the AP (ToDS) or to the client (FromDS).

More Fragments The More Fragments bit is used to indicate if there are additional fragments to be received. If the pacsket has more fragments, the More Fragments bit is set to 1.

Retry If a station has to retransmit a data or management frame, it will set the Retry bit in subsequent transmissions. The receiving station can use the Retry bit's status to find out if the client has attempted to send the packet multiple times.

Power Management Instead of having a special management or control packet to indicate that a station is entering or leaving power-savings mode, the IEEE decided to include a bit in every packet. If a station wants to inform the AP that it is entering power-savings mode, it sets this bit to 1 in a data packet. To leave power-savings mode, a client sends another data packet with the Power Management bit set to 0. This is why cards that support power savings transmit NULL-function data frames occasionally; they want to change their power-savings state, but don't have any real traffic to send.

Many clients will set or clear the Power Management bit in management or control frames to indicate that they are entering or leaving a Power Management state.This is aviolation of the specification and ultimately leads to common power-management incompatibility between clients and APs.

More Data If a station in a power conservation state identifies buffered packets being held at the AP, it will power up its transmitter and issue a PS-Poll frame to the AP. Upon receiving the PS-Poll frame, the AP will deliver pending packets to the client. If pending packets are waiting at the AP for delivery, the AP will set the More Data bit for each delivered packet. Upon receiving a frame with the More Data bit set, the client will send an additional PS-Poll frame until the More Data bit is cleared.

WEP/Privacy Bit The WEP bit originally indicated whether or not a data packet had been encrypted using the flawed WEP algorithm. Since 802.11i was introduced, the WEP bit is also called the *Privacy bit* and is also set on data packets encrypted using WPA/WPA2. Note the Privacy bit is set to indicate that the frame is encrypted; the use of the Privacy bit does not indicate whether the transmitter supports encryption.

Strict Order Bit A transmitter may choose to reorder the delivery of traffic based on a given application's requirements. If a station cannot accommodate the delivery of out-of-order data, it can set the Strict Order bit to force the transmitter to send frames in order. In practice, this field is generally not used.

Duration/ID This field indicates how long (in microseconds) the station that transmitted this packet needs the media *following this packet*. When a station gains access to the media to transmit a data packet, the receiving station can safely acknowledge that packet, without checking to see if the media is available. The reason this happens is because the duration value in the original data packet included the time required for the receiver to acknowledge it. Unacknowledged packets (such as broadcast data packets) set this to 0.

Any value greater than 32,767 microseconds is illegal. This field also serves other purposes. In PS-Poll frames, instead of a duration, it contains the 14-bit Association ID (AID) of the transmitting client.

Finding and Connecting to Wireless Networks

Most of the management packets mentioned previously were related to connecting (or disconnecting) from a wireless network. This section covers exactly what happens when a station is looking for a wireless network.

Locating Wireless Networks

The 802.11 standard provides two different ways for stations to locate APs: passive scanning using beacons and active scanning using probe request/response.

Clients may opt to discover the presence of networks through passive scanning by listening for the presence of beacon frames. Each AP regularly transmits beacon frames for

network management and clock synchronization tasks, as well as to advertise information about the network to scanning clients. The following illustration shows a Wireshark decoding of a beacon. Notice that beacon packets carry around a lot of information, including the supported data rates, operating channel, traffic indication map (TIM), and other network details. The most interesting field in a beacon is probably the *Service Set ID (SSID),* which is the human-readable name of the network. In the illustration below, the network is named "NETGEAR".

The other way for stations to locate networks is through the active scanning model. Beacon packets are analogous to the AP saying "Hi, I'm Linksys" every 1/10th of a second. Probe requests, on the other hand, let clients actively look for networks. Probe requests come in two flavors: directed and broadcast. A *directed* probe request is analogous to a station transmitting a packet that says "Hello, is a network named Linksys nearby?" by including the SSID of the desired network in the probe request payload. A *broadcast* probe request is more analogous to a station asking, "Are *any* networks out there?" by including a zero-length SSID in the probe request payload, indicating a broadcast SSID.

APs respond to directed probe requests only if they are configured with the SSID included in the probe request frame. All networks in the area are supposed to respond to broadcast probe requests. At least, that's the way it is supposed to work. In Chapter 4 of the book, you'll see that vendors have violated this protocol to let users hide their networks, while hackers have developed techniques (the most notable being KARMA, covered in Chapter 5) that respond to all directed probe requests.

Connecting to a Wireless Network

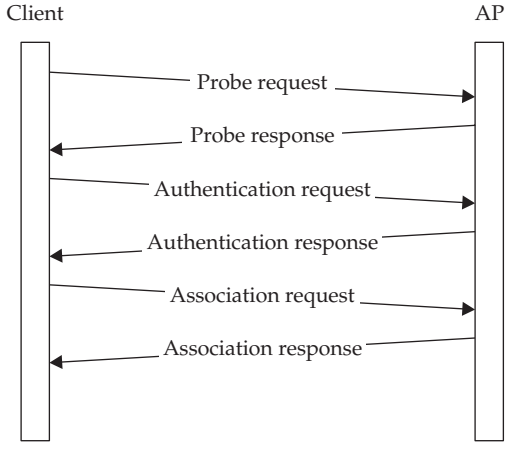
Assuming that a station has found a wireless network that it wants to connect to, what does it do? The first thing it does is send out an authentication request. This authentication request is merely a formality. The original 802.11 standard specified a shared-key authentication scheme (based on WEP) that was supposed to prevent people from connecting if they didn't know the key. Turns out this type of authentication is actually *worse* than no authentication at all. For this reason, almost all networks simply leave it turned off.

So assuming the network is properly configured (and doesn't use the broken shared-key authentication method), the AP replies with an authentication response indicating the station is authenticated. Once this is done, the client sends an association request.

The association request packet is interesting, in one sense, because it is required to have the Service Set Identifier (SSID), or network name, of the network it is associating to. Some networks try to keep this a secret, despite the fact that every client must transmit it in the clear when they connect.

Association requests carry some information useful to the AP. In particular, when a station is associating, it informs the AP what data rates it supports, whether it can handle certain speed optimizations (such as a short slot time), and so on.

Assuming the station successfully authenticated previously, the AP responds to the association request with an association response. The only really new information in an association response is the status code (assumedly successful), and the station's Association ID (AID). The AID is used to identify clients regarding power savings. The entire six-packet exchange is shown here.



WPA/802.11i Background

Once the IEEE realized that WEP was going to need to be replaced, they started task group i (TGi, sexy name, we know). This group was tasked with creating a new set of security protocols to protect 802.11 against all of the things that WEP didn't. The amendment proposed by TGi is referred to as 802.11i. 802.11i was ratified in 2004 and merged into the most recent 802.11 standard 802.11-2007.

The Wi-Fi Alliance, however, didn't want to wait for the amendment to be ratified before delivering a solution to alleviate the failing WEP protocol. In 2003, they created WPA. WPA is a subset of the 802.11i standard that was completed at that time. While TGi had to be concerned with keeping wireless networks secure for many years to come, the Wi-Fi Alliance was more concerned with securing them in the near-term. The 802.11i

amendment specifies two different data confidentiality protocols: the *Temporal Key Integrity Protocol (TKIP)* and the *Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP).* The bulk of the 802.11i standard remains the same, regardless of which confidentiality protocol is used.

TKIP was designed as a software update for legacy WEP hardware. As a result, TKIP uses cryptographic functions that hardware running WEP can compute. Specifically, TKIP uses RC4 (the same stream cipher used in WEP) and a new Message Integrity Check (MIC), called *Michael.* Both of these algorithms are supported by the majority of wireless cards that were designed with the intention of only supporting WEP. By contrast, CCMP uses AES-based cryptographic functions exclusively. While using AES-based cryptography is forward looking, it also requires different hardware in wireless cards and APs to support this CPU-intensive algorithm.

WPA is essentially the draft form of 802.11i, minus the AES-based cryptography. WPA2 was released after ratification of the 802.11i standard and implements all of the mandatory elements of 802.11i. Practically speaking, WPA is 802.11i with TKIP, and WPA2 is 802.11i with TKIP or CCMP. WPA2 is the approved Wi-Fi Alliance interoperable implementation of 802.11i. For simplicity, we will refer to WPA/WPA2/802.11i collectively as WPA unless there is a compelling reason to pin down a specific implementation.

WPA Groundwork

WPA successfully leveraged much of the infrastructure used to secure wired networks onto the wireless world. This immediately improved the security of 802.11 networks tremendously. It also required network administrators to become familiar with many protocols they might not have run into before. This section aims to clear up the mystery behind all of the protocols that WPA sits on top of.

Few people realize that at WPA's core is a protocol with roots going as far back as 1992. This protocol is the Extensible Authentication Protocol (EAP), and in order to understand what WPA does (and the myriad of intertwined protocols it uses under the covers), it helps to study EAP and where it came from.

Technically, EAP didn't become its own protocol until 1998, when RFC 2284 was first published. However, the beginnings of EAP are evident in the Point-to-Point Protocol (PPP) RFCs (see section 7.2 of RFC 1331), where EAP evolved and first came into use.

PPP is a popular protocol used by dialup ISPs. PPP acts a sort of link layer on your phone line between you and your ISP. It allows you to encapsulate more than one network layer protocol, though it is currently used for IP almost exclusively. PPP includes features that you probably never thought you needed, but it addresses issues that had been quite problematic. The most obvious is how do you handle IP address configuration across a dialup link? Your phone line is not an Ethernet cable; you cannot simply send a DHCP request to the broadcast address. PPP addresses this and various other network layer configuration problems through a library of what it calls *Network Control Protocols (NCPs).* The particular protocol used to configure IP over PPP is IPCP, and it is specified in RFC 1332.

As PPP matured, it became clear that a method for authenticating users was required. Initially, the standard specified two ways for users to authenticate: PAP and CHAP. PAP stands for *Password Authentication Protocol,* but it should really be named *Plaintext*

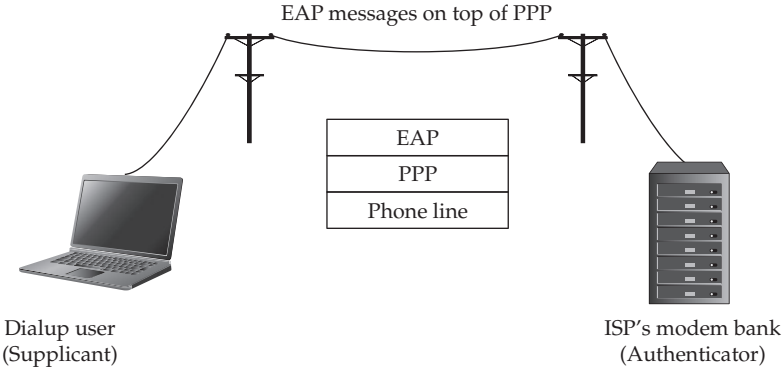
Authentication Protocol. PAP is a euphemism for sending a username and password across the wire in the clear; they had to call it something.

CHAP stands for *Challenge Handshake Authentication Protocol.* CHAP is probably the forerunner to any other challenge-response protocol you might have seen. Basically, the ISP sends you a random challenge, and you compute a hash of a function that takes your password and the challenge as input and sends the hash back. If you know the password, you can compute the hash; if not, you can't. The ISP computes the same password hash, and if your hash matches the ISP's calculation, you must know the secret.

CHAP was a big improvement over PAP (since the password is never sent in the clear), and most dialup users still use either CHAP or PAP. However, the people writing the standard could see that they might be adding more and more authentication schemes as time progressed. Instead of modifying the PPP protocol every time a new authentication scheme was cooked up, they modified it so PPP could handle arbitrary authentication schemes. They did this by creating EAP.

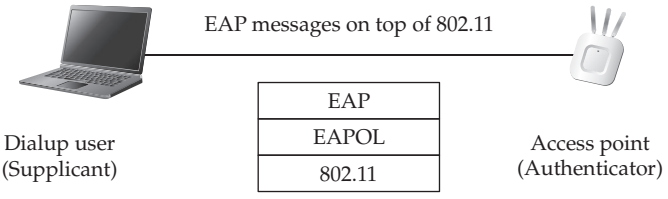
Extensible Authentication Protocol (EAP)

EAP is a very small protocol. When most people think of protocols, their head immediately fills up with layer two protocols such as Ethernet, layer three protocols such as IP, and so on. EAP is very different than the protocols most people are familiar with. It is designed to run directly on top of layer two protocols (such as Ethernet or PPP), but it is not really supposed to carry arbitrary network/transport protocols on top of it. Also, unlike most protocols that run directly on top of the link layer, there is no real addressing scheme. As far as EAP is concerned, there are only two entities: the entity requesting to be authenticated (for example, a dialup user or a wireless client) and the entity on the other end of the link doing the authenticating. EAP has been modified to use the same terms for these entities that 802.1X uses (EAP came first, but the 802.1X terminology became more ubiquitous). Consequently, this means that EAP calls the entity that wants to be authenticated the *supplicant,* and the guy on the other end of the link doing the authentication is called the *authenticator*. The dialup user scenario is shown here.



To return to wireless security for a moment, consider the next diagram. The layer labeled *EAPOL* stands for *EAP over LAN,* which is covered in the 802.1X section. Although

EAP is certainly used in wireless authentication, we are going to continue with the dialup example because that is where the protocols covered in this section actually originated.



Now that you know who EAP exchanges messages with, I'll describe those messages. EAP is deceptively simple; there are only four categories of messages defined in the standard (RFC 3748):

- **Request** Request packets are sent by the authenticator to the supplicant.
- **Response** Responses are sent from the supplicant to the authenticator.
- **Success** Sent by the authenticator to the supplicant to indicate that authentication was successful.
- **Failure** Sent by the authenticator to the supplicant to indicate that authentication was unsuccessful.

The layout of an EAP request/response is shown here.

1 byte	1 byte	2 bytes	1 byte	5 bytes
Code	Identifier	Length	Type	Type-Data

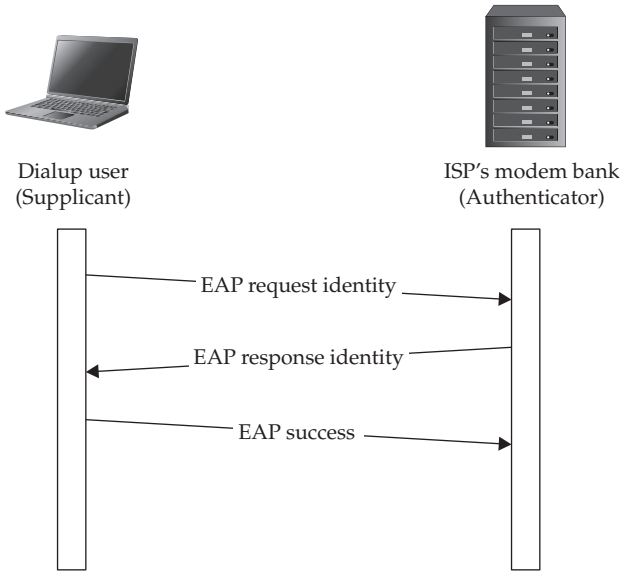
In the packet, the Code field is used to determine if the EAP packet is a request, response, success, or failure message. The Identifier is used as a serial number to pair up responses with requests. Length specifies the size of the packet (in bytes, including the header). The interesting fields are the Type and Type-Data fields.

Type indicates the specific authentication type being used. Only a few of these are defined in the standard. The people who created the EAP standard are *not* responsible for creating specific authentication schemes. Examples of authentication schemes include EAP-TLS, which authenticates the supplicant with the use of certificates, or EAP-MD5, which is a simple challenge-response authentication scheme using MD5. The important thing to remember is that the authentication-specific details of all these messages are buried inside the Type-Data field.

Once EAP authentication begins, any number of request and response packets may be exchanged. The only requirement is that the exchange ends with either a success or failure. Regardless of the authentication scheme being used, an EAP packet with the Code field set to 3 (success) will be sent if authentication is successful.

In the simple exchange shown next, the authenticator asks the supplicant for his name. When the supplicant responds, the authenticator decides to let the supplicant in. This might

seem like a trivial authentication scheme, but it can be useful. For example, some sort of token device might generate the user's identity. The real motivation for this example, however, is to introduce the EAP request identity message.



One of the few authentication types that the EAP standard specifies is the EAP request identity type. If it is hard to conceptualize asking someone to identify themselves as being a reasonable authentication scheme, so be it. Just think of it as the standard overloading of the Type field for a special case. Regardless of whether or not asking someone his or her identity is a real authentication scheme, the standard explicitly covers it, and it is used in the opening phases of almost any authentication scheme. Even if you are going to deny someone access, you at least want to know who they *claim* to be.

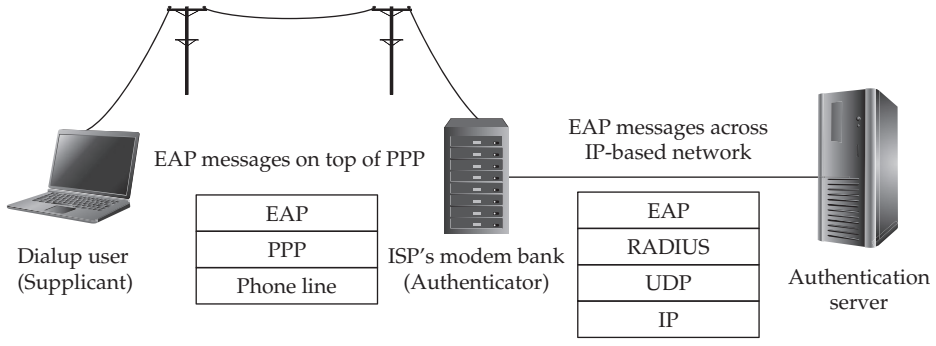
Generally, the identity sent back is a username of some sort. The standard doesn't lay out any hard and fast rules, however. An identity could be a computer name, for example.

The most important thing to remember about EAP is that any authentication scheme can be put on top of it. This allows new authentication schemes to be deployed without disrupting the basic protocols carrying EAP messages around.

Introduction to RADIUS

So far in the dialup authentication example, the user can authenticate to the ISP using any authentication scheme that takes place over EAP. The problem with this model is that the EAP messages are only being passed from the user to the ISP's modem bank (or *point of presence,* as it is called). If the user were authenticating with a username and password, every modem bank would need its own copy of the username/password database. This is

clearly inefficient. A more desirable model is shown next. In this model, the authentication server is a central repository of usernames and passwords.



The protocol that forwards EAP messages from the authenticator to the authentication server is called *Remote Access Dial-In User Service (RADIUS)*. RADIUS was originally defined to solve the username/password database problem when using the built-in CHAP/PAP authentication over PPP. Once EAP was created, and advanced authentication techniques implemented on top of it, RADIUS was modified to transport EAP. This is specified in RFC 2869.

As far as wireless security is concerned, RADIUS is really just a crunchy old protocol being used to transport EAP packets from an access point to an authentication server, and that is how we are going to treat it. This crunchy old protocol happens to run on top of UDP, so you can finally get the authentication packets from the end-users (who have no network layer connectivity) routed somewhere. RADIUS actually contains other complicated features related to accounting and so on, but these features aren't strictly related to wireless security and so will be overlooked for now.

RADIUS terminology, unfortunately, does not match up with that used in EAP or 802.1X. This biggest difference is that RADIUS calls the point of presence (or access point in the wireless world) a *Network Access Server (NAS)*. This is confusing because as far as RADIUS is concerned, the access points and POPs are really clients. EAP and 802.1X both refer to these entities as *authenticators*.

RADIUS resembles EAP in some sense (or more accurately, EAP resembles RADIUS since RADIUS came first). Like EAP, it has four important messages related to authentication:

- **Access-request** Packets sent from the NAS (access point, POP) to the authentication server
- **Access-challenge** Responses sent from the authentication server to the NAS (access point, POP)
- **Access-accept** Sent by the authentication server to the NAS to indicate successful authentication
- **Access-reject** Sent by the authentication server to the NAS to indicate unsuccessful authentication

You can see from the terminology used for the RADIUS messages that RADIUS was designed with CHAP in mind. This is why access-challenge is called access-challenge, not access-response. Nonetheless, there is an orderly relationship between these four RADIUS packet types and EAP. When using EAP over RADIUS, EAP request messages are sent to the authentication server inside access-request messages. EAP responses are sent back to the authenticator (or NAS, in RADIUS speak) in access-challenges.

The original motivation presented for this three-tier authentication architecture was to avoid duplicating copies of a username/password database to all the POPs an ISP operates. A more general consequence of this is that the authenticator no longer has to process (or even comprehend) the authentication scheme employed between the end-user and the authentication server. As shown in Figure 1, consider, in general, what the authenticator is doing now.

The authenticator simply takes the end-user (supplicant's) EAP packets, wraps them up in a RADIUS packet, and sends them to the authentication server. Similarly, when it receives a response for the authentication server, it strips off the RADIUS packet, pulls out the EAP frame, and forwards it on to the client. The authenticator continues proxying these messages back and forth (without having to understand them) until it receives a RADIUS access-accept/access-reject message from the authentication server. The net result of this is that you can drastically change the authentication scheme on your

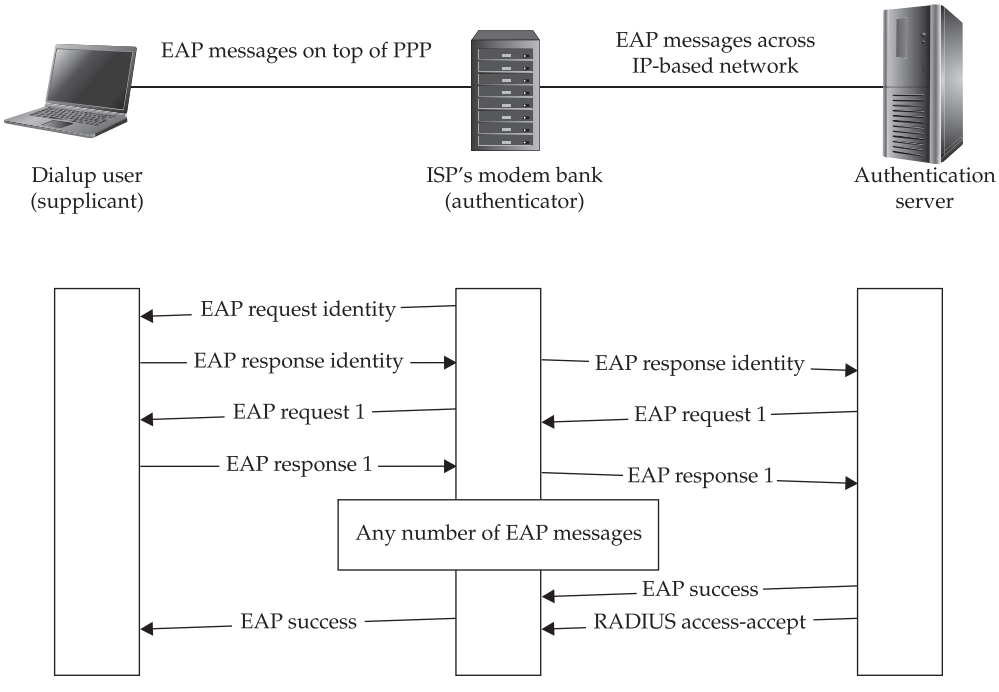


Figure 1 Sample authentication exchange between a dialup user and authentication server

wireless network (or dialup network) without having to modify any settings on your access points/POPs.

Once the ISPs modem bank receives the RADIUS access-accept packet, it knows that the authentication server has authenticated the client, and it is okay to let her connect. Keep in mind that at no point during this exchange did the modem bank have to know what sort of authentication method the dialup user and authentication server employed.

802.1X: Bringing EAP to the LAN

We have been discussing authentication protocols in terms of dialup users for two reasons. One is that all of the protocols mentioned so far were developed in the context of dialup users. The other is that it lets us avoid introducing even more cumbersome terminology.

802.1X is another standard controlled by the IEEE (the 802. is a dead giveaway). In some sense, 802.1X is unique because it is tightly related to EAP. EAP, however, is administered by the IETF and documented in RFCs. The simplest way to describe 802.1X is to say that it wants to bring the EAP-based authentication outlined previously to everyday LAN users. The most straightforward case to consider is Ethernet.

Port-based Access Control

802.1X basically splits every physical Ethernet port into two logical ones: the controlled port and the uncontrolled port. The idea is that *anyone* who plugs into the physical Ethernet port can communicate over the uncontrolled port. There is a catch, however; the only type of communication allowed over the uncontrolled port is related to authentication.

Note

The "X" in 802.1X is always uppercase, a designation by the IEEE to indicate that 802.1X is a stand-alone specification. Amendments to an existing specification (such as 802.11i) use a lowercase letter.

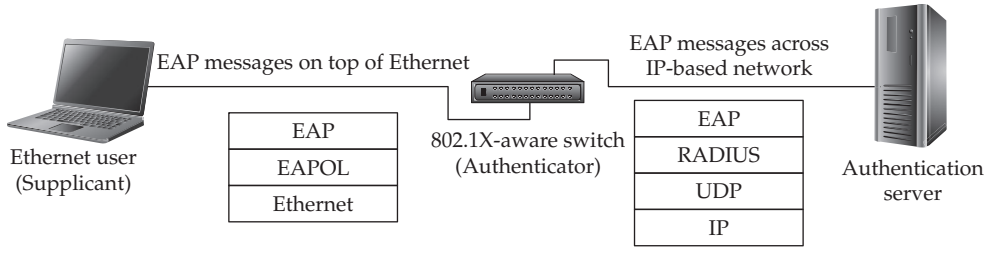
Once a user successfully authenticates (over the uncontrolled port), data is allowed to pass over the controlled port. As a not-so-subtle reminder that IEEE has a lot of electrical engineers, a port that *allows* data to flow across it is called *closed*. This makes sense if you think of the port as a switch in a circuit, but it is very counterintuitive for mortals who spend most of their time a few layers higher in the protocol stack, where the term *closed port* means precisely the opposite.

To recap, when a user plugs into an Ethernet port protected by 802.1X, the only thing the user is allowed to send at first are packets related to authentication (*EAP packets*). Once a user successfully authenticates, she is allowed to transmit normal Ethernet data packets.

EAP over LAN (EAPOL)

We mentioned earlier that EAP typically runs directly over the link layer, but that isn't entirely true. EAP could go directly on top of Ethernet; however, the IEEE decided it would be a good idea to wrap it in something called *EAP over LAN (EAPOL)*. In the most minimalist sense, EAPOL is simply a way to ferry EAP packets from a supplicant to the authenticator. In this case, the supplicant is a user with a laptop plugged into an Ethernet port, and the authenticator is the 802.1X-enabled switch that the user is plugged in to. In practice, EAP is still too simple to go

over the bare metal of Ethernet (or other link-layer protocols such as 802.11). When doing port-based access control, EAP is not designed to address some things that need to be considered. Two of these things are notifying the authenticator that you would like to authenticate and informing the authenticator that you are officially disconnecting. Although this might seem unnecessary in wired networks, it poses more of a problem in wireless settings. As you'll see later, the main reason you need a wrapper for EAP is actually key distribution. The following illustration shows authentication using 802.1X on an Ethernet-based LAN.



The four basic types of EAPOL message are discussed here briefly:

- **EAPOL-packet** The most intuitive type of EAPOL packet. These packets are simple containers for transporting EAP packets across a LAN, for example, from a user's laptop to the 802.1X-enabled switch or access point.
- **EAPOL-start** The supplicant can use this packet to inform the authenticator it wants to authenticate. In many cases, this is unnecessary as the authenticator can sense the supplicant is connected before it transmits an EAPOL-start message.
- **EAPOL-logoff** This message informs the authenticator that the supplicant is disconnecting from the network—also unnecessary in many cases.
- **EAPOL-key** The 802.1X standard provides support for key distribution, which is very important when it comes to securing wireless networks.

802.1X Summary

802.1X defines a way to transport EAP packets across Ethernet and other link-layer protocols to an authenticator embedded in an 802.1X-aware switch or access point. This allows organizations to use their authentication server (most likely RADIUS-based) to authenticate users before they can use an Ethernet port (or, as you will see, wireless link) to transfer data. In addition, 802.1X provides support for key distribution, which is important for securing wireless links.

WPA/WPA2: Putting It All Together

Now that you have seen all of the protocols on which WPA builds, we'll describe what WPA is responsible for exactly. Intuitively, people expect that WPA should encrypt their packets so that "other people" cannot read them, which is true. The question to ask

though is "Which other people"? People outside your organization who are not authorized to use your wireless network? Well, certainly. What about other people who have a legitimate reason to use your wireless network? In other words, should employees be able to read each other's mail as it flies by? Preferably not, but at least this is less of a problem. In security lingo, both of these questions are really questions of *confidentiality*.

The next biggest concern people have is that of *authentication.* You want to ensure that the only people who can connect to your network are ones authorized to do so. Notice that all of the protocols WPA borrows from wired security were focused on authentication, not confidentiality. EAP/RADIUS/802.1X, as they have been described, are concerned with making users prove that they are authorized to use the network. Once they have done that, these protocols have done nothing to encrypt the communication that takes place afterward. Dialup links and Ethernet, even when protected by strong authentication that sits on top of EAP, are *not* encrypted once authentication takes place. WPA/802.11i spends most of its time filling this gap.

There are more subtle security concerns as well. One related to wireless is that of *integrity*. When you receive a packet on your wireless network, you want to be sure that it wasn't modified (or even injected by someone else). Finally, another concern about wireless security is *replay protection.* This was a serious problem with WEP. Even when attackers didn't know the WEP key, they could blindly replay packets and use this to generate traffic, which, in turn, could be used to recover the WEP key. WPA takes strong precautions to address all of these problems.

As it turns out, once you have a strong authentication mechanism in place, solving the rest of these problems is significantly easier. This section first delves into the details of authentication in a WPA environment. Once we cover this, all of the other problems become much more simple to address.

Authentication Using WPA

Having covered 802.1X and RADIUS, the authentication aspect of WPA looks a lot like that of a user plugging a laptop into an 802.1X-protected Ethernet jack. Instead of passing EAP packets (wrapped up in EAPOL) over Ethernet, the packets are passed over 802.11 (still wrapped up in EAPOL). Instead of a switch being the authenticator and authenticating physical ports, the access point now acts as the authenticator and authenticates everyone who associates. Figure 2 shows a simplified association/authentication exchange when using WPA/WPA2 with an authentication server.

Up until this point, everything in the WPA authentication exchange phase directly resembles that of Ethernet when using 802.1X (excluding the 802.11 association requests and responses, of course). This is the point after which 802.11 really starts to dictate what happens and where wireless security starts to diverge from wired.

If the sole thing you are interested in is protecting your wireless network from outsiders authenticating to it, you would be finished. The authentication server would be able to distinguish between legitimate users and attackers, and the AP would only let legitimate users in. In short, this would be a very brief section.

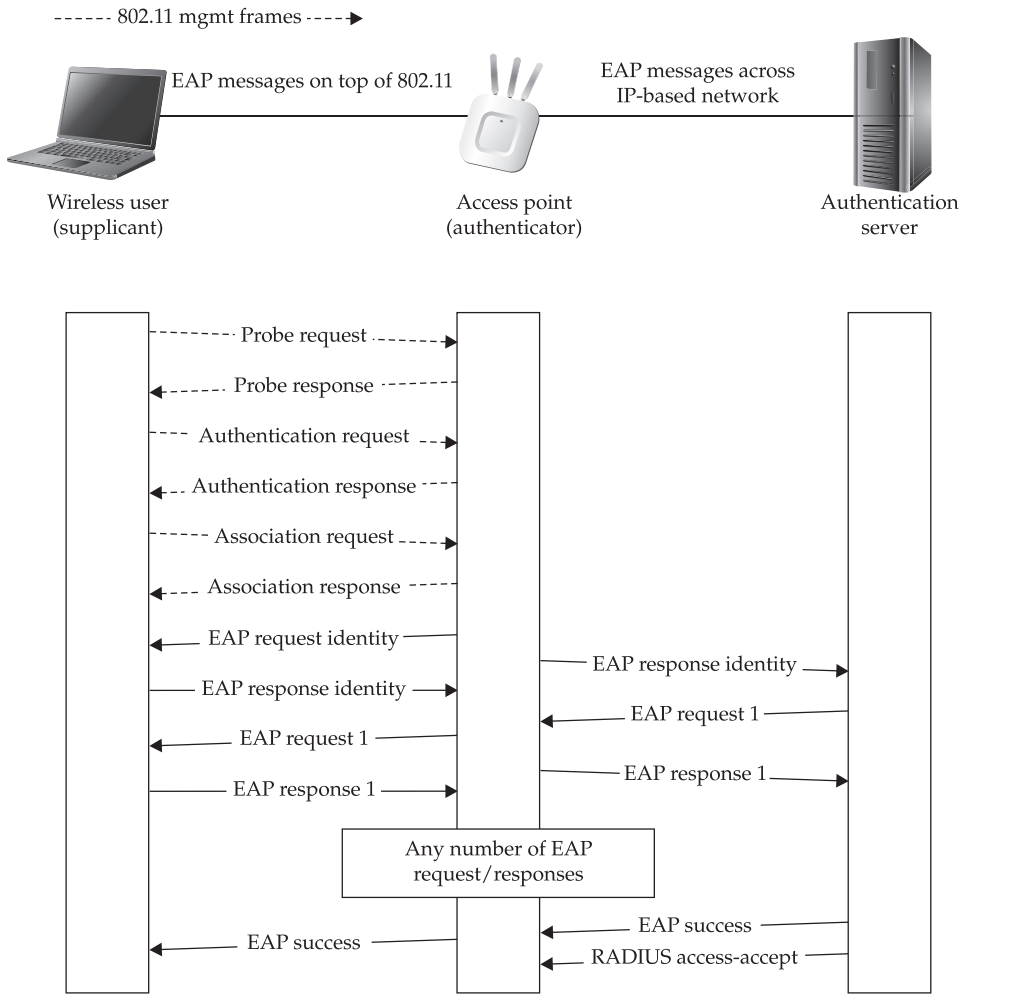


Figure 2 Simplified association and authentication exchange in WPA-protected networks

The problem is that you also want to protect your users from connecting to rogue access points. If the protocol just stopped after authenticating a client, a rogue AP could simply pretend to authenticate the end-user, accepting whatever username/password (or any other authentication credentials) he sent. Another problem that needs to be addressed is ensuring that authenticated sessions aren't hijacked. While it is difficult to hijack someone's authenticated 802.1X Ethernet session (you would need physical access to the wire), no such barrier exists in 802.11.

If the protocol stopped following the EAP Success message, any halfway competent attacker trying to get on the network would just wait for a legitimate user to authenticate,

DoS them off the network, and clone his MAC address. The access point would never know. The solution to both of these problems is to provide a much stronger binding between the AP and client than just a MAC addresses. You need a secret key.

WPA EAP Authentication Requirements

Though so far this chapter has largely been concerned with authenticating dialup users with usernames and passwords, such a simple scheme is unacceptable with WPA. In order to prevent users from being tricked by rogue access points, and to stop attackers from simply stealing the MAC address of authenticated stations, WPA requires that the access point and the user share a secret key. In WPA terminology, this key is called the *pairwise master key (PMK)*. The easiest way to explain what a PMK is and how it is generated is to consider a real-life authentication session.

Figure 3 shows an authentication exchange between a user and an authentication server using *EAP-TLS.* TLS is the successor of the SSL protocol, which is used to authenticate web servers (to you) as well as encrypt subsequent communication (such as your credit card number). When used in secure HTTP exchanges, TLS doesn't usually authenticate users to the server (since users don't have certificates). However, TLS does include support for mutual authentication, which is used in EAP-TLS.

The way EAP-TLS works is that the server presents the client with a certificate, which the client must verify. The client then presents the server with his certificate, which the server must verify. Assuming both certificates check out, the client and server negotiate a session key. Though the details of the negotiation have been left out, conceptually this exchange is fairly simple. Since the client has the server's certificate (and, therefore, the public key), the client generates a random number (the premaster secret) and sends it to the server encrypted with the server's public key. After that, the client and server mix in a few other random numbers and call it a *session key*.

In HTTPS, this session key would be used to initialize a stream cipher, and the rest of the communications over HTTP would be encrypted using it. In EAP-TLS, you are not interested in using TLS to encrypt the 802.11 packets. EAP-TLS uses TLS for three things: to authenticate the server to the client, to authenticate the client to the server, and finally to generate a cryptographically secure session key, which you will use for your own purposes.

Following the authentication exchange, the RADIUS server must inform the AP that the user has successfully authenticated, while also disclosing the derived PMK. The RADIUS server has to do this because the AP has no idea what session key the client and authentication server negotiated within the TLS tunnel. Once the AP has the key, it will install it as the PMK for the client and perform a four-way handshake.

Remember, the AP is not looking into the authentication details of the EAP packets it passes around, and even if it was, it couldn't decrypt the premaster secret used to generate the TLS session key. Therefore, the RADIUS server has to send the AP the PMK that has been negotiated. The AP uses the PMK to ensure that the wireless client it is talking to is the one the authentication server authenticated.

If the authentication protocol used is well designed (such as EAP-TLS), then an eavesdropper should not be able to determine what PMK the client and the authentication server have negotiated. In the case of EAP-TLS, the eavesdropper would have to compromise the private key associated with the authentication server's certificate.

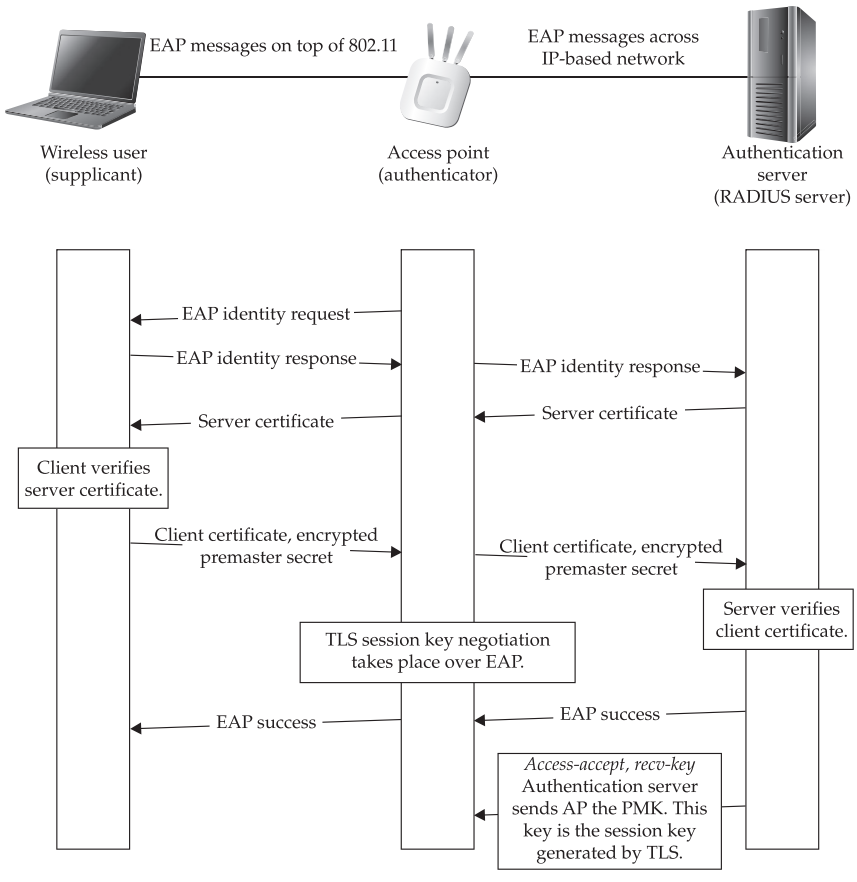


Figure 3 EAP-TLS authentication

At this point, note that key delivery is not something that RADIUS was originally designed to do. Clearly, you need to protect keys as they traverse the network between the authentication server and access point. Currently, this is handled by an extension to RADIUS developed by Microsoft known as the Microsoft Point to Point Encryption protocol (MPPE) and is documented in RFC 2548.

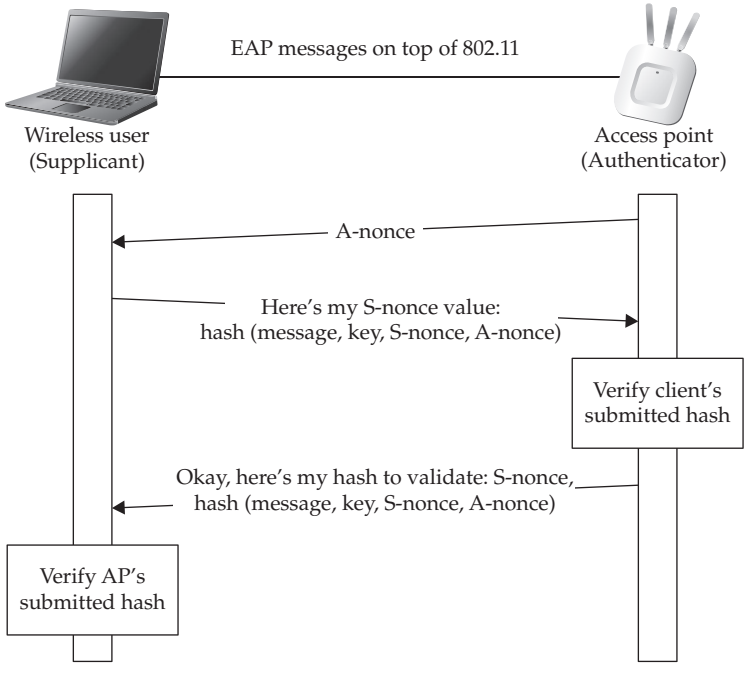
Authenticating the AP to the User, and Vice Versa

At this point, you know how an access point and an end-user can both end up with a dynamically generated cryptographically secure key, called the pairwise master key (PMK), without it being exposed to an attacker over the air. When using WPA in enterprise mode, this key is generated at an authentication server and delivered to the AP over RADIUS. The dynamic generation of this key is what necessitates the use of so many different protocols (EAP, EAPOL, and RADIUS, to be specific). Now that the client and access point both have this key, you are almost finished.

Now the AP must prove to the user that it does, in fact, have the key that the user negotiated with the authentication server. Doing this prevents an attacker from waiting until the user has authenticated to the authentication server and then attempting to launch an attack designed to redirect the user to a rogue access point. Remember, in the previous TLS example the client authenticated the authentication server via his certificate, not the AP. Just because an AP has the same name and MAC address as the one you were just talking to a second ago, doesn't mean it really *is* the same one. This is what makes wireless security so tricky.

Similarly, the AP needs to ensure that the client actually possesses the PMK. Otherwise, an attacker could just let a legitimate user authenticate with the authentication server, wait for the authentication server to send an Access-accept message to the AP, and then disable the client, and steal her MAC address. The possession of the PMK proves the identity of both the client and the AP to each other.

Now, all the AP and the client need to do is convince each other they actually possess the PMK. They also need to do this without transmitting the key over the air. They accomplish this with a simple challenge-response protocol, as shown next. This exchange has been simplified to avoid introducing more terminology.



In the protocol shown in this illustration (which is modeled off the real four-way handshake employed in WPA), the first thing that happens is the AP sends the client a random number. This number is called the *A-nonce* (for *authenticator-nonce*, where a nonce is a number that is not re-used). The client generates its own nonce (S-nonce), and then sends a message to the AP informing it of the S-nonce value. This message contains

a hash of the message, A-nonce, S-nonce, and an expansion of the PMK as inputs. Once the AP receives the S-nonce value, it can compute and verify the hash.

The AP then transmits a message containing a hash of the message, A-nonce, S-nonce, and expansion of the PMK, which the client verifies. If both sides compute the same hashes, they have proven they each possess the PMK without transmitting it.

WPA Authentication Summary

This concludes the detailed explanation of WPA's authentication mechanisms. Basically the authentication server and station dynamically generate cryptographic keys and those keys are delivered over RADIUS to the AP. The AP and the client then need to mutually authenticate each other. As you will see later, the PMK is used to derive a series of other keys, without which WPA could not provide confidentiality or guarantee the integrity of packets.

Confidentiality in WPA

Now that authentication has been covered at length, you know how WPA ensures that users end up talking only to the right AP, and APs talk only to the right users. You also know how unique PMKs are created during the authentication phase and delivered to the AP from the RADIUS server. This is important because confidentiality in WPA assumes that the AP and client already have the PMK in place.

As mentioned previously, WPA/WPA2 describes two data confidentiality protocols, TKIP and CCMP. Though the details of how a packet is encrypted depends on each protocol, for now you are concerned with key distribution and key hierarchy. Neither of these aspects is dependent on the specific confidentiality protocol.

You might think that because both the AP and user now have the PMK, you can just throw it into an encryption algorithm and finally start sending some data. However, you aren't done dealing with keys. Although having one key at the AP and another at the client is a good start, when it comes to cryptographic keys in WPA, the more the merrier.

The 802.11 Key Hierarchy

Although the PMK is in place at the AP and at the client, you are going to perform a few logically distinct cryptographic operations. The most obvious of these is computing a Message Integrity Check (MIC) over all the packets and also encrypting them. These are two different operations when using TKIP, and it would be a poor design choice to use the same key for different things.

The 802.11 key hierarchy defines a way that the PMK can be used to create a set of temporary keys. These keys are called *transient keys*. When TKIP is being used, five transient keys are created: one for encrypting network traffic, two for packet integrity, and two other keys that we won't delve into here.

When using CCMP, the data encryption/integrity roles are actually combined (through the use of a clever AES mode of operation), and only four keys are needed. For simplicity, let's refer to the entire set of temporal keys at once. This set of combined keys is called the *pairwise transient key (PTK)*.

The PTK is recomputed every time a station associates and is recomputed after every 65,536 encrypted packets are sent by the transmitter (either the supplicant or the AP). This helps ensure that the derived PTK is unique, even if the PMK is the same. When using WPA in pre-shared key mode (also known as *personal mode*), this is important because, in this case, the PMK is usually constant across all users (the WPA specification does say how to use per-user pre-shared keys, but it is rarely implemented or used).

Generating the PTK from the PMK is relatively straightforward. All you are doing is expanding one random 256-bit number (the PMK) into a 512-bit number for TKIP keys or a 384-bit number for CCMP keys. You also need the derived numbers to be a function of some information exchanged during the four-way handshake. The PTK is defined as follows:

PTK = PRF(*PMK*, "Pairwise Key Expansion", *MAC1, MAC2, nonce1, nonce2*)

This function is a pseudo-random number function (PRF) defined in the 802.11i standard. It takes the PMK, as well as various nonces and the MAC address of the AP and client as input, and returns a 512-bit number for TKIP or a 384-bit number for CCMP.

You might be wondering what that "Pairwise Key Expansion" string is doing. The PRF family of functions is used elsewhere in the standard as well. This motivated the IEEE to use constant strings (such as the "Pairwise Key Expansion" string) as input to ensure that even if the same pseudo-random number generator is called using the same nonce and MAC addresses, but for different purposes, different output is produced. Attention to small details like this is critical when designing security protocols.

The Four-way Handshake

The four-way handshake was alluded to in the section on authentication. In that case, it was simplified to provide only authentication. In reality, the four-way handshake serves two purposes. One purpose is to prove that both the AP and the client possess the PMK. This feature was explained earlier in "Authentication Using WPA." The second purpose is to provide the relevant values that need to be plugged in to create the PTK. In particular, these values are the nonces used to derive the PTK mentioned previously. The full four-way handshake is shown in Figure 4. The fact that this handshake is doing two things at once throws people off, so it is covered in some detail.

The first message in the four-way handshake consists of the AP sending the client its nonce. In practice, it is probably a number that will very likely never be used again, such as a large random number. Once the client has the A-nonce, she has all the information needed to compute the PTK.

With the A-nonce, the client chooses her own S-nonce and derives the PTK. She takes the S-nonce and puts it in a EAPOL-key message that is sent back to the AP. Using one of the keys contained in the PTK, she also computes a MIC over this packet. By computing this MIC, she proves to the AP that she knows the PMK. If the client *didn't* know the PMK, she couldn't have derived the PTK. Without the PTK, she could not have computed the MIC.

Once the AP receives the second message, he knows the S-nonce chosen by the client. The AP then derives the same PTK and computes the MIC over the received packet. If the computed MIC doesn't match, then the client did *not* derive the PTK and, therefore, is lying

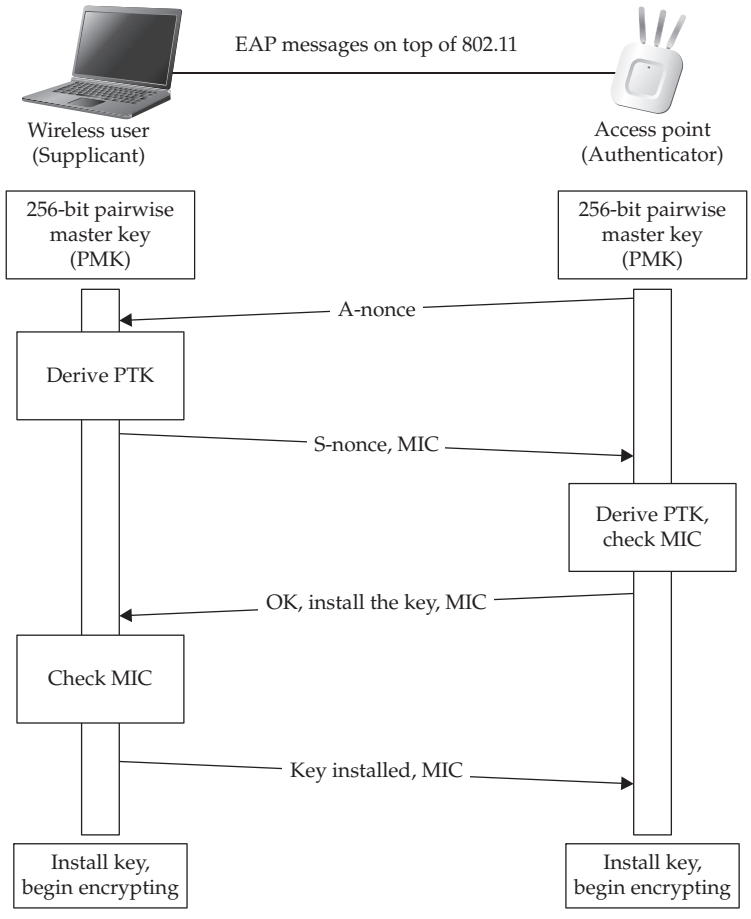


Figure 4 The four-way handshake

and will not be allowed access. Assuming the MIC the client sent matches, the AP responds with a message informing her that she has successfully authenticated and to go ahead and install the key. This message is protected by a MIC that the client must verify to ensure the AP has possession of the PMK. Assuming the MICs on both ends are correct, the AP and the client have proven possession of the PMK (via successful derivation of the PTK), and they have authenticated each other.

Once the client verifies the MIC on the AP's message telling her to install the keys, she sends back a response, and the AP and client install the keys and start to encrypt the data.

Confidentiality Summary

Once all of the work of authenticating to the network (and the network to the client) is completed, encrypting packets is relatively straightforward. Since authentication has

already happened, the client and the AP share a secret key (the PMK). This key is expanded from one random 256-bit value into multiple key values that represent the pairwise transient key (PTK). Contained within the PTK is an individual key, called the temporal encryption key, which is used to initialize encryption via TKIP or CCMP.

Integrity in WPA

The next security problem that WPA solves is that of integrity. After authenticating, the AP and client end up with a shared PMK. After association, 802.11 creates temporary keys (the collection of which are referred to as the pairwise transient key, or PTK). One of the individual keys contained in the PTK protects the integrity of messages. Before talking about how this works, let's consider what would happen if WPA didn't provide any sort of integrity check.

Many people think that because their messages are encrypted, they don't need to worry about them being modified. Since an attacker can't read them, how could he meaningfully modify them? Although an attacker cannot read them, he still might be able to flip random bits and gum up the works. He might not know what he is changing, but he could still try to modify messages.

A bigger problem is that even if an attacker doesn't know the key, he could still try to inject data into your network. With no integrity check, consider what happens when an attacker crafts an arbitrary packet and sends it directly to you (bypassing the AP). When you get his packet, you'll decrypt it, and since the packet wasn't properly encrypted to begin with, you'll transform the payload into a stream of nonsensical bits in the process. The attacker can't choose what the bits will look like when they're decrypted (because he doesn't know the key), but he can try to inject *something*. The question is, how do you know that this nonsensical packet wasn't sent by a legitimate network user?

Well, you might say that no real user would send you a stream of bits that isn't a valid IP packet (which is mostly true), but you don't want your wireless device driver sitting around trying to guess if a packet was sent by a real user or an attacker trying to inject something (even if just random noise). The way this is accomplished is with a Message Integrity Check (MIC).

Message Integrity Check

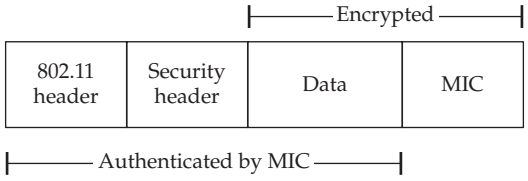
In 802.11 (and other IEEE 802 standards), the term *Message Integrity Check* is used instead of the more accepted *Message Authentication Code* because the acronym *MAC* was already taken. MIC is IEEE-speak for what everyone else calls a Message Authentication Code.

Conceptually, a MIC is straightforward. Because the goal of the MIC is to prevent the data from being modified in transit, the sender computes a hash over the data and sends the hash as well. Of course, if the attacker can modify the bits of the message, she could modify bits of the hash. Instead of having the sender just compute the hash of the data, she computes a hash of the data plus a secret key. The key used in WPA is the temporal integrity key (contained in the PTK). Mathematically, this can be written as

$$\text{MIC} = \text{hash}(\textit{packet, temporal integrity key})$$

When using TKIP, the hashing algorithm is called Michael. CCMP uses the Cipher Block Chaining Message Authentication Check (CBC-MAC) hashing algorithm. A diagram of

a packet processed by WPA is shown next. In this diagram, the 802.11 header contains all of the normal stuff you would expect to find (addresses and so on). The security header field contains parameters specific to the TKIP or CCMP protocol, including the packet initialization vector. After that, you have the actual payload of the packet, followed by the MIC.



Caution Michael MIC protects only a subset of the 802.11 header, most importantly, the source and destination address.

Notice that the MIC itself is also encrypted. When the recipient of this packet receives it, she will have to decrypt the payload and the MIC. She will then compute the keyed hash over the header and payload and compare it to the MIC. If the MIC doesn't match, she'll discard the packet. This prevents an attacker from modifying packets as they fly by (how could he make the MIC match his modifications without the MIC key?), and it also stops unauthenticated users from injecting packets (how would they know what key to use to compute the MIC?).

Replay Protection in WPA

The last feature that WPA provides is replay protection. Compared to the other problems WPA addresses, replay protection might seem a little less important. You already know how to stop attackers from getting on the network, reading packets, modifying packets, and injecting their own packets. The last thing that an attacker might try (perhaps out of frustration that she couldn't do anything else) is replay packets that a legitimate client already sent.

Imagine an attacker replaying a packet that tells the bank to transfer money to her account; if successful, she could increase the amount of money transferred by replaying the legitimate transaction one or more times. That is obviously a pretty cooked-up example, but it does provide some motivation to prevent it. In reality, the biggest threat that replaying packets poses is the ability to generate illegitimate traffic on a LAN.

Although this might not sound like that big of a threat, the lack of replay protection contributes significantly to the speed at which a WEP key can be recovered, so it is still important to consider.

Adding Serial Numbers to Packets

The solution to preventing replay attacks is simple. Simply add an incrementing number to every packet exchanged between the client and access point. In TKIP, this field is called the *TKIP Sequence Counter (TSC),* and in CCMP, it is just called the *Packet Number (PN).* Since these numbers perform the same logical function, I'll call them *serial numbers.*

Whenever a client associates to an AP, it starts counting the packets it sends. The first packet gets serial number one. These numbers restart whenever the client negotiates a new PTK (by disconnecting and reassociating, or through regular key rotation). This serial number is embedded in the TKIP or CCMP header of every packet.

If the AP wants to determine if a packet has been replayed, it just checks to see if the serial number is fresh. Intuitively, you might think that the AP could expect the serial counter to always increase by one; but in reality packets get dropped, and the protocol needs a little slack to compensate. Currently, the standard says that any packets that arrive with a serial number equal to the expected serial number, minus a small window (currently 16), will be processed. Any packets with a value less than this, which would be an old replayed packet, are dropped. This small window allows the protocol to deal easily with the occasional lost packet, while still preventing an attacker from replaying old data.

This serial number is unencrypted in the TKIP/CCMP header. A smart attacker who wanted to replay a packet could simply modify the serial number and retransmit it, right? That would be possible, except the serial number is used as input for the decryption process. By trying to reinject an old packet with a modified serial number, the packet will fail to decrypt.

Devices processing WPA-protected packets will perform two checks, either of which should cause the packet to be dropped. First, they will check the ICV (checksum). Unless the attacker does some magic, the ICV will be wrong, causing the packet to get dropped. The next check is the cryptographic hash introduced by WPA. Without the key, he can't make the cryptographic hash (MIC) work out, and it will be dropped.

Summary

This Bonus Web Chapter covered the basic types of packets used in 802.11 networks. The unique features as well as the motivation for their inclusion in the 802.11 MAC were covered briefly as well. The methods that wireless clients use to locate and associate to networks were discussed in some detail.

When trying to understand all the features that WPA/WPA2/802.11 provides, determining the responsibilities of the other protocols it depends on can be a stumbling block. These protocols were presented in a historical context that helps explain why WPA works the way it does. The protocols that WPA builds on (EAP, 802.1X, and RADIUS) were covered in enough detail to relate them to wireless security. All of the features that WPA/802.11-2007 brings to wireless networks were also reviewed. These include robust authentication, integrity, confidentiality, and replay protection schemes.