8 Chapter

Access Control: IEEE 802.1X, EAP, and RADIUS

This chapter introduces some of the protocols that are central to the new security solutions. One of the most basic functions needed for security is access control and the new security solutions are built around a standard, IEEE 802.1X, which is specifically designed to implement access control. This chapter starts by describing the need for access control and then shows how the control techniques developed for dial-up modem pools have been reused in conjunction with WI-Fi LANs. As we will see, the combination of IEEE 802.11, IEEE 802.1X, EAP, and RADIUS provide a solution scalable from home networks all the way to large corporate networks.

Importance of Access Control

Access control is one of the most important elements of security. The object of security is to separate the world into "good guys" and "bad guys." It follows that you cannot achieve security unless you have a mechanism to perform this separation. That mechanism is **access control**.

On the surface, maintaining control is straightforward. All situations have the following elements:

- An entity that wants to have access—the **supplicant**
- An entity that controls the access gate—the **authenticator**
- An entity that decides whether the supplicant is to be admitted—for now, we will call this the **authorizer**

Suppose a visitor knocks on your front door and your child opens it (with the security chain on). The visitor is the supplicant, your child is the authenticator, and you are the authorizer. Only if you say it's okay will your child take off the security chain and let the visitor in (don't you wish you really had such power!). If you answer the door personally, you take the role of both authenticator *and* authorizer.

The steps involved in access control follow a similar pattern:

1. Authenticator is alerted by the supplicant.
2. Supplicant identifies himself.
3. Authenticator requests authorization from the authorizer.
4. Authorizer indicates YES or NO.
5. Authenticator allows or blocks access.

These steps work to control access; but as we discussed in earlier chapters, if the supplicant wants to come and go repeatedly without going through this procedure each time, he needs to obtain some sort of token that proves that he has been authorized. In the case of a corporation, for example, that might be a swipe card that opens the employee entrance door.

So if access control is really this simple, why devote a whole chapter to it? Well, the reality is that while the concept and goals of access control are simple, designing a system that is immune to attack is very difficult. Most of the access control systems dealing with people are trivially easy to fool by an intelligent con man. How many of us have left our swipe card at home one day and, upon arriving at work, just walked in behind another employee? For Wi-Fi LANs, we can't allow even the slightest flaw in the access control method, or else hacker tools will appear on the Internet within months. Getting it right is hard.

This chapter focuses on the three protocols that are used to implement access security in WPA and RSN:

- IEEE 802.1X
- EAP: Extensible Authentication Protocol
- RADIUS: Remote Authentication Dial-In User Service

The first two protocols are mandatory for WPA and RSN. RADIUS is the method of choice for WPA and is an option for RSN.

There is much confusion about IEEE 802.1X and what it does. Because it is difficult for customers to fully understand all the elements of security, vendors tend to talk about IEEE 802.1X as if it were the entire security solution for Wi-Fi LANs. In reality, as we will see shortly, IEEE 802.1X is only a small part of the solution, albeit an important one. IEEE 802.1X is the foundation of both WPA and RSN.

IETF Standards

Many of the standards in this chapter and in Chapter 9, "Upper Layer Authentication," were developed by the Internet Engineering Task Force (IETF), an organization that is completely different from the IEEE (although both often involve the same people). All the most basic protocols used on the Internet, starting with the Internet Protocol (IP) itself, have been defined by the IETF. The organization, which operates more on technical consensus and less on formal voting, creates documents called RFCs, short for "Request for Comments." The RFC number for EAP is RFC2284, for example. Despite the title, most RFCs are quite stable and not subject to much change. Perhaps these should transition to NMCTs (No More Comments Thank you), but this would not be in the spirit of continuous technical review, which the IETF encourages.

The stability of RFCs allows vendors to implement and deliver products. New ideas in the IETF are floated using *draft* documents, which are circulated for discussion. Rather than having a number, these drafts have a name incorporating the subject and main author. For example, "draft-haverinen-pppext-eap-sim-03.txt" describes draft three of a proposal written by Henry Haverinen to use EAP with GSM phone systems using a SIM smart card. Proposals that become group work items use the generic "ietf," such as "draft-ietf-pppext-eap-ttls-00.txt," which describes how to use EAP in conjunction with Tunneled TLS authentication.

Many drafts are dropped due to lack of interest, but those that get support from the group eventually move on to become RFCs. This is relevant to EAP because, at the time of this writing, most of the new EAP methods were in the form of draft IETF submissions. In addition, a revised version of EAP was in the works (draft-ietf-pppext-rfc2284bis-02.txt) and expected to supersede the current version. The revision, of course, does not change the existing protocol, but extends and clarify its capabilities. By the time you read this book, this draft might have become a new RFC. All the RFCs and current drafts are publicly available from www.ietf.org.

Before we look at IEEE 802.1X, let's take a diversion and look at the history of dial-in modem support. "Why now?" you may say. The fact is that the main protocols of EAP and RADIUS were both developed in the context of dial-in access.[1] It turns out that dial-in access control is organized in a very similar way to IEEE 802.1X, which is why the same protocols, EAP and RADIUS, can be applied to both. By reviewing the dial-in case first, you will find that the WPA and RSN cases make more sense.

Authentication for Dial-in Users

Millions of people use dial-up access for Internet connectivity every day. Each person's computer is configured with the phone number of the Internet service provider (ISP), a user name and a password; management of the connection is done automatically. Behind the scenes, the majority of these connections use a protocol called Point-to-Point Protocol (PPP), reflecting the fact that the connection is just that—a point-to-point connection between two modems that between them provide an unformatted byte-by-byte link. PPP converts the unstructured modem link into a packet-based environment suitable for transporting the IP packets. Importantly, it deals with initial handshaking during which the two ends can negotiate a common feature set. It also has a mechanism for user authentication that we are interested in here. The original PPP (RFC1661) described simple methods to authenticate the user that are still implemented by many ISPs today. When you initiate dial-up access on your computer, you may see a little box on the screen saying, "authenticating…" while this occurs. You might also be asked to enter a password during this phase. These events typically happen during the PPP negotiation.

PPP has two authentication methods described in the original standard; but by current opinion, neither method is considered very strong. In fact, the simplest method, PAP, which is still used by many ISPs, actually sends the user name and password in clear text so any snooper on the link can steal it. The other popular method, CHAP, uses a challenge response mechanism, somewhat similar to the original WEP method. This is much better than PAP but still not considered very strong.

In the dial-up case, the authentication method doesn't need to be very strong. Generally, the worst that happens if someone steals your password is that they get free access to the ISP for which you may be paying. In addition, dial-up lines are much harder to intercept than LANs. However, in some situations a stronger

1. Strictly, EAP was developed to support Point-to-Point Protocol (PPP), which is very widely used in dial-up networks but also has other applications.

authentication method is needed and the fact that PPP specifies only two weak methods presents a problem. Any new PPP authentication methods would have to be registered with the IANA (Internet Assigned Numbers Authority), and this would create a problem for existing deployed systems that are already "PPP compliant."

To solve this problem, IETF members decided that a more *extensible* method was needed for PPP. Therefore, the option of EAP was added alongside PAP and CHAP. EAP allows full authentication to be delayed until after the preliminary PPP link is established. RFC2284 "PPP Extensible Authentication Protocol (EAP)" describes how this modification is applied and used with PPP; it says nothing about IEEE 802.1X or wireless LAN (neither of which existed when RFC2284 was written). Recognition of LAN applications is one of the changes proposed in the draft update (draft-ietf-pppext-rfc2284bis-02.txt). This will include references to IEEE 802.1X and IEEE 802.11 in the EAP definition.

The intent of EAP is to enable the use of an authentication algorithm between the supplicant and the authorizer. EAP is designed to allow different types of authentication methods to be used —that is why it is called extensible. The object is to enable the supplicant to prove his identity to the authorizer. Many methods allow mutual authentication so both parties prove their true identity to the other.

It is common in dial-up networks to have a modem pool in each local phone area to provide cheap access, often called a **point of presence** (POP). However, the service provider doesn't want to keep a copy of their user database at every POP; they want a central database. This creates a three-party situation that is very similar to that of a corporate Wi-Fi LAN. Using the terminology in our introduction, the user is the supplicant, the POP is the authenticator, and the central database is the authorizer. The protocol used between the POP and the central database to get permission to allow a dial-in user access to the network is called RADIUS (Remote Access Dial-In User Service). We look at RADIUS in more detail later in this chapter.

The organization of a typical dial-in network is shown in Figure 8.1.

Three parties are shown in Figure 8.1:

- **Users** (supplicants)
- **Network access server** (NAS), located in the POP; authenticator
- **Authentication server** (AS), which can be located centrally; authorizer

Note that the term "authorizer" is not an official term, but one that we invented in the introduction to aid in understanding the roles of the parties. From here on, we use the term "authentication server" rather than authorizer because this is most widely used to describe the authorizing entity.

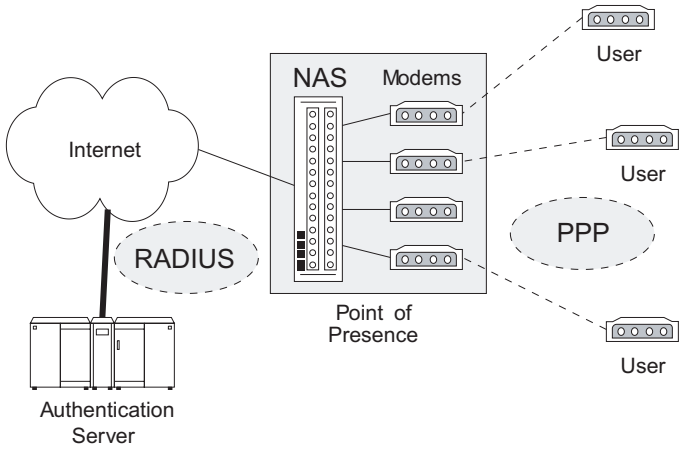

Figure 8.1 Organization of Dial-in Network

EAP allows a flexible approach so arbitrary and complicated authentication protocols can be exchanged between the supplicant and the authentication server. To allow this, RADIUS has been extended to enable EAP messages to be forwarded by the NAS. The NAS acts as a sort of middleman in the authentication process, just relaying EAP messages between the supplicant and the server until the authentication process completes. When the authentication server makes a decision, the result gets sent to both the NAS and the supplicant in RADIUS/EAP messages. This enables the NAS to either allow access or disconnect the unauthorized user.

This three-party model is in use in thousands of POPs around the world (although few use EAP at this time). This is relevant because the situation in corporate Wi-Fi LAN looks rather similar to Figure 8.1. The supplicant is the Wi-Fi user that wants network access. The equivalent of the NAS is the access point, and there is an authentication server that controls the authorization process. The difference is that IEEE 802.11 provides a structured packet network so that PPP is not needed. In the next section we see that the role of IEEE 802.1X is to provide a similar access control function to that performed by the NAS in Figure 8.1.

IEEE 802.1X

IEEE 802.1X is very simple in concept. Its purpose is to implement access control at the point at which a user joins the network. It divides the network universe into three entities along the lines we discussed in the previous section:

- Supplicant, which wants to join the network
- Authenticator, which controls access
- Authentication server, which makes authorization decisions

The point at which a user connects to the network is called a **port**. A network may have many ports; for example, in a switched LAN hub,[2] each Ethernet connector would be one port. There is a one-to-one relationship between a supplicant and a port, and each port has an associated authenticator to control its state. There is a many-to-one relationship between the ports and the authentication server. In other words, a single authentication server is usually responsible for many ports, each with its own controlling authenticator.

Although wireless is our primary concern, IEEE 802.1X was not originally designed with wireless communication in mind. In fact, work was started on IEEE 802.1X before the first version of IEEE 802.11 was completed in 1997. The opening paragraph of IEEE 802.1X says:

> IEEE 802 Local Area Networks are often deployed in environments that permit unauthorized devices to be physically attached to the LAN infrastructure.

Note that the word "physically" implies a wired connection. The original thinking behind IEEE 802.1X was to protect ports such as might be found on a switched Ethernet hub. The idea was to prevent *just anyone* from connecting to the network by plugging an Ethernet cable into a hole in the wall and, instead, require that a potential user's identity and authorization status be checked. As IEEE 802.1X moved toward completion, people recognized that the same principle could be extended from wired ports to wireless connections. Cisco incorporated the concept into its products first, and the approach was adopted for RSN in IEEE 802.11i and, subsequently, by the Wi-Fi Alliance for WPA.

The main point of providing port security is to protect network connections where those connections might be accessible in a nonsecure area, such as a lobby or conference room. For most corporations this is a small number of ports. But for wireless, it is potentially every connection, because the nature of wireless makes almost all links publicly accessible. This is why IEEE 802.1X is so appropriate for IEEE 802.11.

---

2. Because each port must act independently, IEEE 802.1X can only be supported by a LAN switch, not a conventional shared LAN hub.

IEEE 802.1X in a Simple Switched Hub Environment

Before looking at its application in WPA/RSN, let's get an overview of what IEEE 802.1X does in a simple switched hub environment. Control is based around the concept of a switch on each port that is normally open (no connection). The switch is closed only when the supplicant is authorized. As shown in Figure 8.2, the hub ports are all disconnected initially. If anyone plugs in, he doesn't automatically get network access. It is important to note that it would be most unusual for the switches to be implemented as actual physical contacts. The switch here is only logical, like software or logic gates. When the switch is "open," data packets are not forwarded to or from the port. When it is closed, they are sent. The Ethernet port remains electrically active all the time.

One of the obvious problems in the diagram shown in Figure 8.2 is that it doesn't provide any way for the devices to ask the switched hub for permission to connect. It's like forgetting to put a doorbell on your front door. No one can ring to ask to come in. Remember that each port has an authenticator that is responsible for opening and closing the switch, so IEEE 802.1X provides a way to talk to the authenticator even when the switch is open.

This is like the security guard at the front door of an office building. When you arrive you are not allowed in, but you are allowed to talk to the security guard to ask for entry. In the terminology of IEEE 802.1X, the **authenticator** has control over the **port state** (whether the switch is open or closed), as shown in Figure 8.3. Here we see that the device on port 0 has been accepted and is connected to the network; another device is in the process of requesting access to the authenticator on port 1 but does not have access yet. The protocol used to communicate between the supplicant and the authenticator is based on EAP.

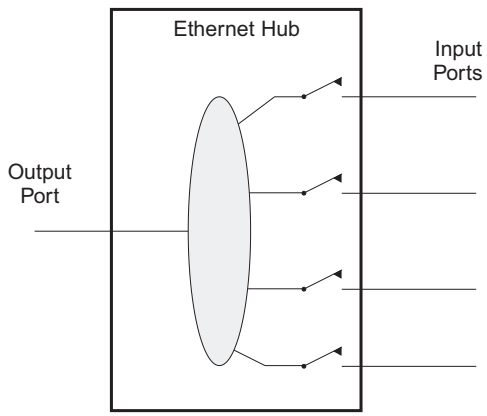

Figure 8.2 Initial State of IEEE 802.1X Switched LAN Hub

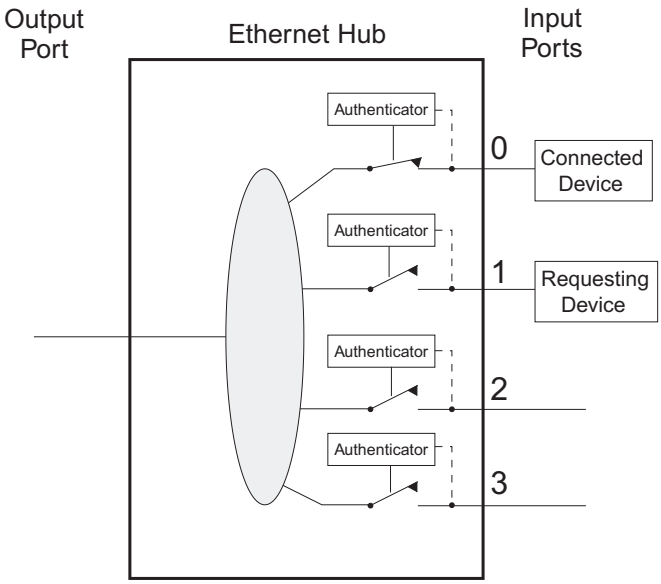

Figure 8.3 Role of IEEE 802.1X Authenticator

In Figure 8.3, it looks as if the authenticators are making the decisions about who is allowed access. In reality, the decision to admit or reject an applicant is usually based on an authentication database controlled and managed by an administrator. For this, the authenticator must communicate to an authentication server in order to get the answer "accept" or "reject" when a supplicant applies to join the network.

The authenticators in Figure 8.3 act like a security guard that has no authority. For every person who comes to the door, the guard has to call upstairs to the boss to find out whether it is OK to allow access. In a small system, the authentication server (the boss) could reside right in the switched hub and would simply have a list of users allowed access. Typically, the list of users would be configured by the system administrator in advance. This approach is impractical except for the smallest networks because each hub would have to be configured separately. Therefore, the authentication server is typically located at some central place in the network, as shown in Figure 8.4.

In Figure 8.4, all four authenticators for the hub shown communicate with the authentication server. In practice, the same would be true for all hubs on the network so the authentication server could be making decisions for many thousands of ports (hopefully not at the same time).

Although we have used a picture of a multiport switched LAN hub in these examples, the IEEE 802.1X standard is really only concerned with one port at a

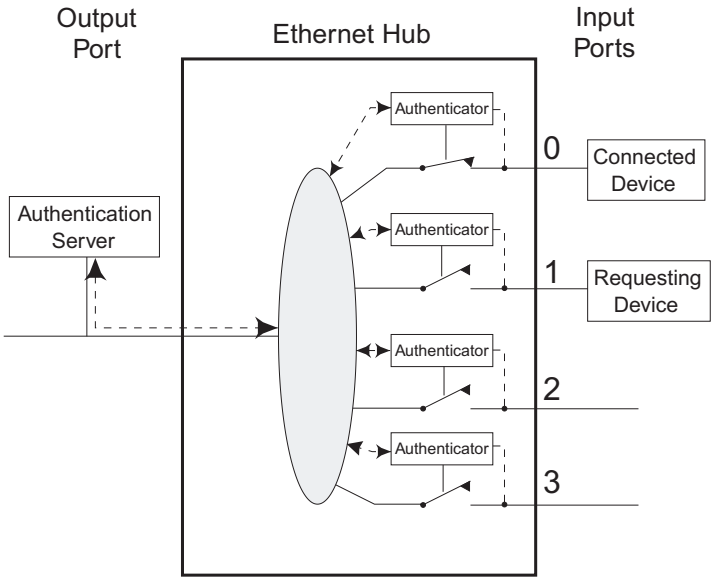

Figure 8.4 Role of Remote Authentication Server

time. Each port has its own state independent of any others in the box. Figure 8.5 shows a picture extracted directly from the IEEE 802.1X specification illustrating the relationships among the entities.

Figure 8.5 again shows the three players: supplicant system, authenticator system, and authentication server system. The supplicant is the device that wishes to get connected. Note that the switch connects through to "services offered by the authenticator's system." Usually we assume this means "connected to the network," but it could be some other service. PAE means port access entity, the full name for a port.

Figure 8.5 includes a reference to a higher-layer protocol between the authenticator and the authentication server. The EAP protocol is used by the various parties to communicate with each other. EAP messages go between the supplicant and the authenticator. The authenticator may also forward them to the authentication server as part of the process of authorization in a similar way that the NAS does in a dial-in network. If the authentication server is in a remote location, these messages need to be sent over a network using some higher-layer protocol. This is where RADIUS can be called into action to transfer the requests over an IP network. RADIUS was designed to perform this job in the dial-up user case, and now we can reuse it for IEEE 802.1X support. WPA specifies RADIUS for this purpose, although other protocols are also possible.

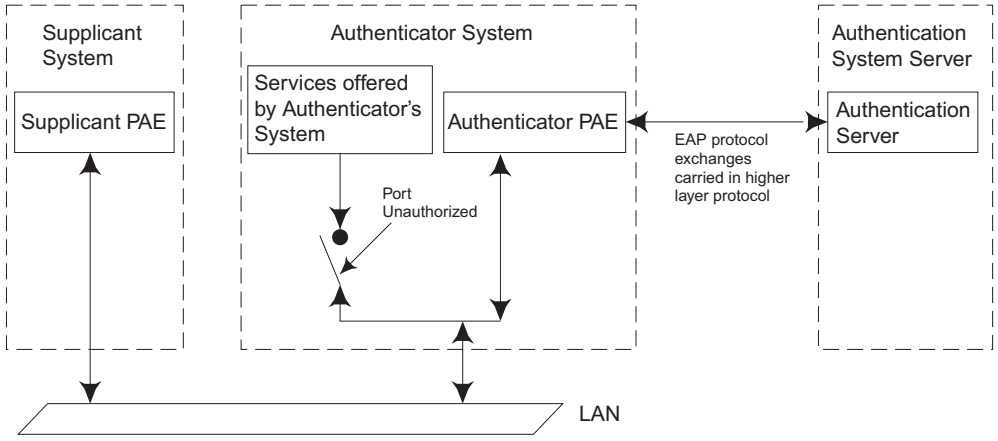

Figure 8.5 IEEE 802.1X Model As Shown in IEEE Standard

One of the differences between dial-in networks and IEEE 802.1X is that, with IEEE 802.1X, there is no need to use PPP because IEEE 802 LANs are designed to send data packets. However, it is still necessary to have some sort of protocol so the receiver can identify the information, and that protocol is called EAPOL (EAP over LAN). As described later in this chapter, EAPOL has several types of messages. Apart from the one that delivers the EAP messages, there are several additional ones that are useful for actions like attracting the attention of the authenticator (the doorbell analogy). Also there is a message for transferring key-related information.[3] WPA and RSN use a similar message in the process of establishing an encrypted link (see Chapter 10).

IEEE 802.1X in Wi-Fi LANs

Given that IEEE 802.1X is designed to control individual LAN ports, how does it map to the wireless case in which there is a single access point supporting many devices? We have to treat each wireless connection between a mobile device and the access point as if it were an independent connection. In effect, we replace the physical connections of a switched LAN hub with logical connections formed by the wireless communications.

In the context of IEEE 802.1X, each mobile device is a supplicant wanting to be provided with the services of the access point (which typically means connection to a wired network). To accomplish this, the access point must create, for

3. This message has no use on wired 802 LANs because they do not support cipher suites.

each supplicant it encounters, a logical port complete with an authenticator. Each authenticator is responsible for controlling access for the mobile supplicant to which it has been assigned. Along with the logical port and authenticator, there is also a logical control switch. As you would expect, the control switch starts in the open position.

A new wireless device, acting as a supplicant, has to apply for access by sending messages to the authenticator, which controls its connection inside the access point. All this is done in software. There is no physical authenticator or switch, so the number of IEEE 802.1X entities in operation is the same as the number of associated mobile devices, regardless of how many that might be (Figure 8.6).

It is a common misconception that IEEE 802.1X is only relevant to big corporate environments in which there are dedicated authentication servers. However, in practice, the authentication server could be a simple process inside the access point—just a list of user names and passwords, for example. This means that the same principles of IEEE 802.1X that apply for huge networks can also apply to home networks. If the authentication server is built inside the access point, there is no need to use RADIUS because the authenticator and authentication server do not need to talk over the network; they are in the same box! However, in this case the number of supported authentication methods would be limited to those selected by the equipment vendor.

So far we have mostly talked about IEEE 802.1X in the context of access control. This has been described as a sort of one-time operation: The supplicant requests access and the authenticator grants it after referring to the authentication server. This may be sufficient for dial-up access or for Ethernet LAN ports because there is a physical connection for each supplicant and it is very hard for an attacker to take over that connection once it is authorized. Clearly the same is not true for

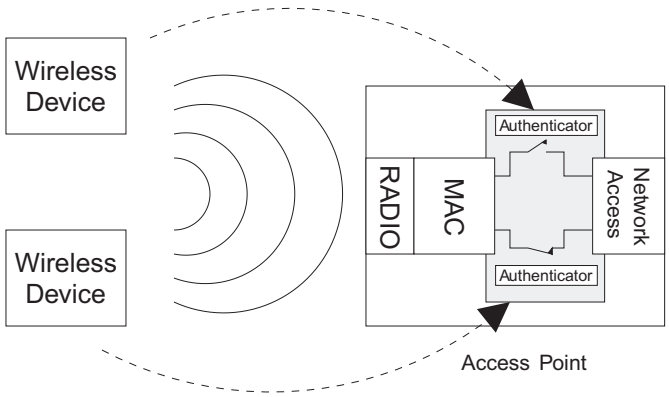

Figure 8.6 Logical IEEE 802.1X Ports in an Access Point

Wi-Fi LANs. Without protection, it would be trivially easy for an attacker to wait until a valid user was granted access and then start using the connection by stealing his identity. Therefore, for Wi-Fi LANs, we have to bind the authorization to a mechanism that prevents this type of session hijack. This is accomplished by incorporating message authentication (integrity) as part of the authorization process. We must be sure that both the access point and the mobile device have their secret keys in place by checking message authenticity and that they have turned on encryption before granting access to the network. This important difference greatly complicates the process and resulted in some minor changes to IEEE 802.1X to ensure synchronization of the process. A new standard, IEEE 802.1AA, is being developed at the time of writing to update IEEE 802.1X partly as a result of its application to IEEE 802.11i. To explore the way that synchronization is achieved, see Chapter 10.

EAP Principles

In some ways EAP performs the role of an actor's agent. When an actor is looking for work, the agent takes her to a movie director and introduces them. The agent sits back while the actor and director talk about the job, but jumps in again at the end to close the deal.

EAP has a set of messages that it uses to make the introductions and to close the deal. These are used with all upper-layer authentication methods.[4] EAP also allows two parties to exchange information that is specific to the authentication method they want to use. The content of these authentication-specific methods is not defined in EAP. In fact, they can be completely proprietary authentication methods or newly invented ones. EAP's ability to handle part of the communication in a standardized way and part in a specific way is the key to its extensibility. We refer to these authentication-specific messages as "middle messages" because they occur after the introduction and before the closing.

Quite a lot of these middle messages can be exchanged before the authentication is completed. The reason why EAP is extensible is that the details of these special messages are left to other RFCs to fill in. For example, there is an RFC saying how to use Transport Layer Security (TLS) over EAP; another (draft) says how to use Tunneled TLS (TTLS) over EAP, and so on. It also means that if you invent a new method later on, you can write a new draft called "mymethod over EAP"; and if it becomes popular, other people can implement it on existing systems.

4. Upper-layer authentication methods are discussed in Chapter 9 and include methods such as SSL, TLS, and Kerberos V5.

RFC2284 (EAP) is a very short document as these things go. In fact, not counting references, acknowledgments, definitions, and so on, it is only nine pages long. RFC2284 (EAP) specifies that four types of message can be sent:

- **Request**: Used to send messages from the authenticator to the supplicant
- **Response**: Used to send messages from the supplicant to the authenticator
- **Success**: Sent by the authenticator to indicate access is granted
- **Failure**: Sent by the authenticator to indicate access is refused

Note that these messages are described here in terms of the authenticator. However, in the IEEE 802.1X scenario, the authenticator forwards the messages on to the authentication server, most likely using RADIUS. In this case it is the authentication server that generates request, success, and/or failure messages and the authenticator just relays them to the supplicant.

Request and response messages are further subdivided using the EAP Type field. The Type field indicates what information is being carried in the EAP message. The first six message types are defined in the standard; all the others are reserved for specific authentication methods. The most important predefined type is Identity (type value 1). Typically, this is used as part of the EAP introduction phase: the message **EAP-Request/Identity** is sent by the authenticator to a new supplicant. The supplicant replies with the message **EAP-Response/Identity** containing its user name or some other identifier that will be understood by the authentication server.

Type numbers higher than 6 are not defined by RFC2284 (EAP), but they are issued (uniquely) by IANA for each new authentication method that is introduced. Some are even issued for vendor-proprietary methods. The type number for TLS, for example, is 13, which means that all EAP-Request and EAP-Response messages with this type field contain information that is specific to the TLS upper-layer authentication method.

The use of the Type field is a bit inconsistent. For the most part, it indicates the authentication method. But in a few cases, it defines a special-purpose message. For example, a message with a type value of 2 is called a **notification** message and is used to send some user-displayable text. This could be anything from "Please enter your password" to "Prepare to meet thy maker"—it really doesn't matter. The message is intended to appear on the screen of the user's system (although few systems actually support this). A message with a type value of 3 is called a NAK and is used when a request is made for an authentication method that is not supported. If an EAP request with type TLS is sent to a peer that doesn't support TLS, it can respond with a Type field of NAK.

Type value 1 **Identity** could be considered a special-purpose message or it could be considered a very simple authentication method. Under IEEE 802.1X, this request is often the first thing sent and the supplicant will reply with a response message giving its identity information. Originally this was treated as a special type to be used prior to the main authentication phase. However, this has been subtly changed in the revised EAP draft (while remaining compatible with the previous version). The simplest authentication exchange could go:

- EAP-Identity request (from authenticator)
- EAP-Identity response (from supplicant)
- EAP-Success (from authenticator)

Here the device has been "authenticated" on pure trust: "I choose to believe that you are who you say with no proof." Or perhaps proof is available by some other means. For example, the identity might be generated by a smart card that changes every second, synchronized to the authentication server.[5] This type of null authentication can be used with simple wireless LAN networks that have preloaded secret keys (called **preshared** keys) and then rely on the encryption to prevent unwanted communications.

Because the EAP-Identity exchange can be considered a complete authentication method by itself, when you do the identity exchange followed by another method such as TLS, you are really running two authentication methods in sequence. This concept of **serial authentication** has been generalized in the new EAP draft, which simply lists the EAP-Identity message as a basic authentication method and then says that you are allowed to run as many authentication methods in sequence as you wish prior to the final EAP-Success or EAP-Failure message.

This ability to run multiple authentication methods in sequence can be exploited in new approaches that allow the client to authenticate the network before revealing its identity. One approach, PEAP (Protected EAP), is discussed in more detail in Chapter 9.

EAP Message Formats

All EAP messages have a similar basic format, as shown in Figure 8.7. **Code** is one byte indicating the type of message:

- Request (01)
- Response (02)

5. This is often referred to as a *one-time password*.

Figure 8.7 EAP Message Format

- Success (03)
- Failure (04)

Identifier is a value in the range 0–255 and IEEE 802.1X indicates that it should be incremented for each message sent. When a response is sent, the identifier is set equal to that in the request. This helps for checking which response goes with which request. **Length** is the total number of bytes in the EAP message (including Code and so on). It is a 16-bit value. Finally, **Data** is the actual request or response data being sent.

We have already discussed the Success and Failure packets. These messages are short and contain no data. One of these messages is used at the end of the authentication process to signal the result. Because the Success and Failure are common across all authentication protocols, intermediate devices (such as the access point) can detect when an authentication completes without understanding all the details of the authentication method. The access point should wait for the RADIUS Accept message before making any decision about access rights.

The details of the authentication method are sent in the request and response messages. These have an extra field called Type. The format of an EAP-Request or EAP-Response message is shown in Figure 8.8.

You can see the Type field, which is used to identify the request or response. The Type field is essential to separate all the different authentication methods. In fact, it is the key to the *extensibility* of EAP. Each new authentication method is assigned a unique value so the system knows whether the request contains information relevant to, for example, TLS or PEAP.[6]

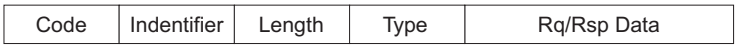

Figure 8.8 EAP-Request/Response Message

6. These authentication methods are described in Chapter 9.

EAPOL

The EAP RFC does not specify how messages should be passed around. It does not, for example, specify transport over the Internet using IP. In fact, EAP *is not a LAN protocol at all* because EAP was originally designed for use with dial-up authentication via a modem. So if we are going to get EAP messages passed around our network, we have to find a way to *encapsulate* the EAP messages during their journey. IEEE 802.1X defines a protocol called **EAP over LAN** to get EAP messages passed between the supplicant and the authenticator.

IEEE 802.1X provides the description for EAPOL. It describes frame formats for Ethernet (IEEE 802.3) and token ring LANs but not for IEEE 802.11. If you just wanted to encapsulate the EAP message, you could prepend an Ethernet MAC header on an EAP message and send it over the LAN. But the IEEE 802.1X committee decided to add a few more useful messages and fields while it was defining EAPOL. Not all EAPOL frames actually carry EAP messages; some are used to perform administrative tasks. The five types of EAPOL messages are:

- EAPOL-Start
- EAPOL-Key
- EAPOL-Packet
- EAPOL-Logoff
- EAPOL-Encapsulated-ASF-Alert

We won't deal with the last message type here. It is connected with what we think is the rather dangerous idea of allowing an unauthorized device to send management alerts to the system. This message is not used by WPA/RSN.

EAPOL-Start

When the supplicant first connects to the LAN, it does not know the MAC address of the authenticator. Actually it doesn't know whether there is an authenticator present at all. To help get things going, IEEE 802.1X defines a message called EAPOL-Start. By sending this message to a special group-multicast MAC address reserved for IEEE 802.1X authenticators, a supplicant can find out whether an authenticator is present and let it know that the supplicant is ready. In many cases the authenticator will already be notified that a new device has connected from some hardware notification. For example, a hub knows that a cable is plugged in before the device sends any data. In this case the authenticator may preempt the Start message with its own message. In either case the authenticator sends an EAP-Request Identity message using the EAPOL-Packet frame (perhaps twice, if both send the initial message at the same time).

EAPOL-Key

Using this message type, the authenticator sends encryption (and other) keys to the supplicant once it has decided to admit it to the network. Of course it is necessary to encrypt the keys themselves before sending them, and IEEE 802.1X does not specify how this is done.[7] In fact, IEEE 802.1X offers little help when it comes to combining encryption with the authentication process. This was a major obstacle that had to be overcome in the WPA/RSN network design. Chapter 10, "WPA and RSN Key Hierarchy," outlines how a slightly modified EAPOL-Key message is used to establish encryption keys and also to validate that both sides have correct keys before allowing access.

EAPOL-Packet

This EAPOL frame is used for sending the actual EAP messages. It is simply a container for transporting an EAP message across the LAN, which was the original objective of the EAPOL protocol.

EAPOL-Logoff

This message type indicates that the supplicant wishes to be disconnected from the network.

For reference, the format of an EAPOL frame for use by Ethernet is shown in Figure 8.9. All of the packet types listed above fall into this format.

- The protocol version is always 1 (this could change in the future).
- The packet type number indicates start, key, and so on.
- For some message types, no further information is needed and the packet body length is set to 0 (and the body is omitted). However, if there is a packet body, such as an EAP message, its length and data are added on as appropriate.

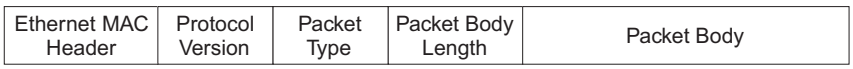

Figure 8.9 EAPOL Frame Format

7. This is rectified in IEEE 802.1AA.

Messages Used in IEEE 802.1X

Messages must pass between three parties: the supplicant, authenticator, and authentication server. IEEE 802.1X uses EAP, or more specifically, EAPOL, to pass these messages between the supplicant and the authenticator. Let's start by following through the sequence of events when a new supplicant arrives.

Authentication Sequence

An outline of the authentication sequence is shown in Figure 8.10. When a supplicant wants to connect, it must first attract the attention of the authenticator. In most cases the authenticator is alerted by the connection process. It might be that the act of plugging in the cable or, in the case of wireless, associating with the access point is enough. Otherwise, the EAPOL-Start message can be used.

The authenticator first responds with an EAP-Request/Identity message. This is a standard EAP message that is equivalent to shouting, "Who's there?" The authenticator is allowed to skip this step if it knows the identity of the supplicant by some other method. The supplicant must respond with an EAP-Response/Identity message. This raises an interesting issue because so far the supplicant can't be certain whether the authenticator can be trusted, especially in a wireless network. Suppose the authenticator is a rogue access point set up by an attacker. The supplicant might not want to reveal its identity at that time and uses a pseudonym instead.

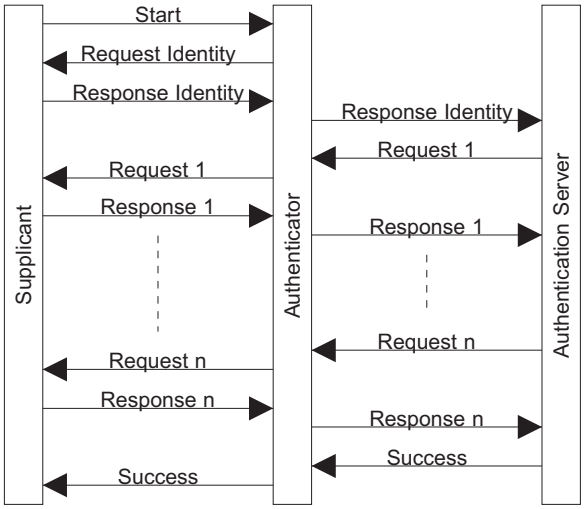

Figure 8.10 EAP Message Flow

Some schemes support the use of pseudonyms[8]; but, for the moment, let's assume the supplicant is not shy and is prepared to send its identity to the authenticator.

So far all the messages we have discussed have gone between the supplicant and the authenticator (in IEEE 802.11, this would be the mobile device and the access point). The authentication server has not been involved until now. It is important not to waste the authentication server's time until the supplicant has shown that it actually speaks IEEE 802.1X by responding to the first EAP-Request. Having obtained the identity of the supplicant, the authenticator needs to contact the authentication server to find out whether this supplicant is to be allowed in. The authentication server can't make this decision until it has verified that the supplicant really corresponds to the identity it has given. This is the whole point of authentication. To avoid the need for the authenticator (in the access point) having to know all the authentication methods, the EAP messages used for authentication are passed directly to the authentication server.

In effect, during this phase the supplicant and the server are talking directly. In our earlier office building analogy, the security guard has opened the door and asked the person's name, but not let him in yet. Then the guard calls his boss and says, "Can we let Harry Acker in?" The boss runs through a set of questions, which the security guard asks Harry one by one. The guard passes the answers back to the boss. The guard just relays the questions and answers and, in the end, the boss makes a decision on entry. Note that during this phase, the guard might not understand the purpose of the questions as in the following scenario:

Harry to guard: Hello, can I come in?
Guard to Harry: Who are you?
Harry to guard: I'm Harry Acker.
Guard to boss: Harry Acker wants to come in.
Boss to guard: Ask him whether the oak tree is a mammal or a marsupial.
(Guard asks Harry)
Harry to guard: It is a marsupial.
(Guard tells boss)
Boss to guard: Don't let him in.
Guard to Harry: You can't come in.

Note that the questions and answers relayed by the guard made no sense to him but clearly enabled the boss to make a decision that the guard then put into effect.

8. For example, PEAP.

During the authentication process, the authenticator takes a quick look at each EAP message that is passed between the supplicant and authentication server. It is watching for certain messages that it understands. In particular, it is looking for an EAP-Success or an EAP-Failure. It must wait until the authentication server indicates whether the supplicant has been accepted or rejected. A RADIUS message provides the indication when RADIUS is being used.

Implementation Considerations

So much for the theory, but where does IEEE 802.1X reside in real systems? For most Wi-Fi LANs, the logical place to put IEEE 802.1X is in the access point. In fact the close coupling between IEEE 802.1X and key management makes it hard to place it anywhere else. There were proposals in the standards work that would allow the key management and wireless access point functions to be separated so IEEE 802.1X could be placed on a separate access box to which the access point was connected. This approach was not adopted for WPA/RSN.

It is possible to build wireless LANs without an access point using IBSS or ad-hoc mode. In this case, it is necessary for every mobile device to have both a supplicant and an authenticator operating in parallel (see Chapter 13).

Some operating systems such as Microsoft Windows XP have support for IEEE 802.1X supplicants built in. When configuring the clients, you only need to enable IEEE 802.1X-based authentication and choose the authentication method. Of course the choice of authentication methods may be limited and you may have to install additional software to get the method you need. In older operating systems, IEEE 802.1X is probably not built in and you will need to install special drivers from the manufacturer of the Wi-Fi equipment you are installing. In all cases, supporting generic IEEE 802.1X is not enough for Wi-Fi LAN. There are other special requirements of WPA/RSN related to key management that must be built into the IEEE 802.1X implementation. In general the manufacturer of the adapter card provides all the necessary hooks and drivers to implement this extra stuff when the operating system does not. You should confirm that when a vendor advertises RSN or IEEE 802.11i compatibility that it does properly integrate with the operating system you intend to use. Systems labeled "Wi-Fi WPA" are likely to have the necessary software and will have been tested for interoperability with other vendors.

IEEE 802.1X can also be used in embedded mobile devices such as mobile phones or PDAs. In this case, the operating system may not be visible to the user. If the device supports IEEE 802.11i RSN or WPA, all the integration issues should be taken into account in the device. However, you will probably have little or no flexibility on the authentication method available. Be sure to find out what authentication method is used on such a device and confirm that your authentication server can support it.

As a final note, remember that IEEE 802.1X itself does *not* define the way that EAP messages are passed between the authenticator and the authentication server. However, it strongly hints that RADIUS is a good way to go in IP networks. It even includes an annex section outlining how RADIUS might be used. RADIUS has already been mentioned and is covered in the next section. Remember that RADIUS is needed only if the authentication server is remote to the authenticator. IEEE 802 deals with LAN protocols generally and is applicable to LANs regardless of whether they use TCP/IP. IEEE 802.1X does not specify RADIUS because it is based on IP packets, which are part of the TCP/IP protocol family. In reality, IP networks are by far the most common, but this was not always the case and IP still isn't used everywhere. Here we assume that you *are* using an IP network and we focus on RADIUS where there is a network connection between the authenticator and the authentication server. WPA goes further and defines RADIUS as a mandatory implementation choice to help ensure interoperability.

RADIUS—Remote Access Dial-In User Service

Although RADIUS is not specifically part of the IEEE 802.11i standard, many practical corporate implementations use it to communicate between the access point and the authentication server. Small office or home installations are very unlikely to use RADIUS because the authentication server is probably inside the access point. So what exactly is RADIUS? Is it a protocol or type of product? You will often hear the term RADIUS server. Is this something you can buy, or can you go to your PC shop and say, "I'd like to buy a RADIUS server please"?

The exact definition of a RADIUS server is a source of confusion. There are companies that make and sell authentication servers. You can make your own authentication server by installing a commercial software package on a conventional PC. However, there is no standard definition for the features of such servers. Some authentication servers are dedicated to specific authentication methods. Others may have special capabilities such as redundant or distributed operation. A **redundant server** has standby units that take over seamlessly if the primary server fails, and a **distributed server** has many servers operating in different locations while it keeps a common authentication database updated and consistent between all sites.

RADIUS defines two things. First, it defines a set of functionality that should be common across authentication servers. Second, it defines a protocol that allows other devices to access those capabilities. When we talk about a RADIUS server, we are talking about that subpart of the authentication server that supports the RADIUS capabilities; and when we talk about RADIUS, we are generally referring to the protocol used to talk to the server.

RADIUS is specified by the IETF and is designed for use with TCP/IP type networks; it assumes that devices use an IP network to talk to a RADIUS server.

As with many aspects of the Internet, the capabilities and needs of systems are continuously evolving, and RADIUS has been stretched and bent by various additions over the years. Therefore, when you buy an authentication server that includes RADIUS capability, you need to ensure that it supports any new bells and whistles that you might need. For example, EAP over RADIUS (RFC 2869) is needed for IEEE 802.11i RSN security, but it was not included in the original RADIUS specification (RFC 2865). RADIUS allows the definition of vendor-specific attributes for special features that the server might provide. One such special feature, Microsoft's MS-CHAPv1/v2 authentication method, is used widely and has almost become a standard requirement.

The first RADIUS RFC (specification) was RFC2058, issued in 1997, although this was superceded almost immediately by RFC2138. In 2000 RFC2138 was further updated and replaced by RFC2865. As noted at the start of the chapter, one motivation behind RADIUS was the support of dial-in modem pools. An ISP might want to provide dial-up access over a substantial area or even nationwide. Customers don't want to pay long-distance phone call charges, so the ISP has to set up a modem pool in each local phone area so users can connect cheaply (or for free). At each modem pool site is a dial-in server that answers the calls, authenticates that the user is a valid customer, and then runs the PPP to allow connection to the Internet. The problem is that each modem pool server needs to know all possible valid users in order to perform the authentication step. The motivation for RADIUS is to have a central authentication server that knows all the customers and allows the modem-pool servers to forward the authentication information to the central site for checking. In RADIUS terms, the modem pool server is the NAS (network access server) and the authentication server (AS) is the RADIUS server.

The analogy with a Wi-Fi LAN is clear. In our case the access point is like the NAS. There may be many of them dotted about the place, and we don't want each one to have to know the authentication database. We can use the RADIUS server, as was intended, to provide centralization of the authentication decisions. If you want to read the specifications, the ones that are relevant to WPA/RSN are:

- RFC2865: Remote Authentication Dial-In User Service (RADIUS)
- RFC 2866: RADIUS Accounting
- RFC 2867: RADIUS Accounting for Tunneling
- RFC 2868: RADIUS Authentication for Tunneling
- RFC2869: RADIUS Extensions
- RFC 3162: RADIUS over IP6
- RFC 2548: Microsoft Vendor-Specific RADIUS Attributes

RFC2869 is relevant because it contains information on how to use EAP over RADIUS. Note also that at the time of this writing, there is a draft RFC potentially updating RFC2869 This draft is called Draft-aboba-radius-rfc2869bis-10: RADIUS Support for Extensible Authentication Protocol (EAP). This draft update recognizes that EAP is now also used in IEEE 802.1X applications in addition to PPP dial-up modems. The original RFC gives examples based on PPP, but this has been generalized in the update.

RADIUS Mechanics

This section reviews how RADIUS works at the protocol level. The basic message set for RADIUS is deceptively simple. Most of the complexity lies with messages called attributes.

Core Messages

The core protocol of RADIUS is very simple. There are just four relevant messages:

- Access-Request (NAS → AS)
- Access-Challenge (NAS ← AS)
- Access-Accept (NAS ← AS)
- Access-Reject (NAS ← AS)

In the WPA/RSN case, the access point is the equivalent of the NAS and AS is the RADIUS authentication server.

These four messages reflect the fact that PPP, the dial-in modem protocol, has two options for authentication: PAP and CHAP. PAP is a simple user name/password approach. CHAP requires that the server send random data called a **challenge**, which the dial-in system must encrypt and return for checking. Let's consider how this works with dial-in.

First we'll tackle the PAP case, as shown in Figure 8.11: The user dials in and the NAS answers and indicates that it is using PAP authentication. The user's system then responds by sending the user name and password for the account. The NAS now sends an Access-Request message to the RADIUS server containing

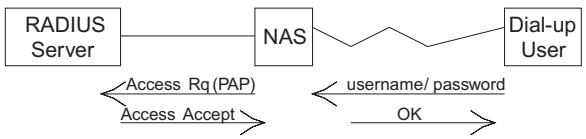

Figure 8.11 PAP Operation

the user name and password information.[9] The RADIUS server responds with either Access-Accept or Access-Reject and the NAS acts accordingly. This is a very simple approach and, of course, it is subject to a wide range of attacks. The worst part is that the password is sent unencrypted over the phone link so anyone monitoring the link can copy it. It is about as secure as one of those little padlocks that come with cheap suitcases—just pretend security, really.[10]

CHAP is a little better, and makes an attempt at secure authentication, as shown in Figure 8.12. Rather than sending the password unencrypted across the phone link, the user sends only its user name to the NAS. The NAS now needs to respond with a challenge. To get the challenge data, the NAS could send the user name to the server using an Access-Request, whereupon the server would send the challenge data using Access-Challenge. However, in most implementations the NAS avoids disturbing the server and generates the challenge by itself, as shown in Figure 8.12. The challenge is passed back to the dial-in user's system, which is required to encrypt the challenge with the password and send it back. Finally the NAS is able to send the challenge, response, and identity to the AS, indicating that it is using CHAP.

This approach means that the password is not sent unencrypted; it also provides some liveness because the challenge changes on each access attempt. However, it is still subject to dictionary attack because both the unencrypted and encrypted versions of the challenge are accessible to an attacker.

Partly because of the dictionary attack weakness, Microsoft implemented a modified version of CHAP called MS-CHAP that is now used widely in corporate dial-up pools. Microsoft "standardized" their attribute definitions through RFC2548, Microsoft Vendor-Specific RADIUS Attributes.

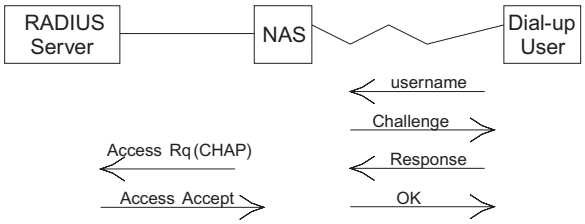

Figure 8.12 CHAP Operation

9. Actually, it sends an encrypted version using the "hiding" algorithm in RFC2865.

10. In defense of PAP, the threat model for PPP assumed that the telco wire was secure. This is generally a pretty good supposition, in which case there is nothing so bad about passwords in the clear.

RADIUS was specifically designed with two PPP authentication scenarios in mind: simple password request PAP and challenge response CHAP. In WPA/RSN, we need to use RADIUS in conjunction with a security protocol that is state of the art—far more complex (and secure) than the simple PAP and CHAP methods. To do this, we need to change the purpose of some of the messages in RADIUS. For example, to support EAP we will use the access-challenge method, not as a challenge, but as a way to send EAP requests and responses. The good news is that RADIUS is flexible enough to accommodate these changes. One of the reasons it is flexible is because of its use of attributes.

Core Message Format and Attributes

Although essentially only four messages are used for authentication via RADIUS, the meaning of the messages can be changed extensively through different message attributes. Figures 8.11 and 8.12 show how the Access-Request message can mean three different things at different times. The attributes the message carries are different in each case. The main body of the RADIUS message is composed of a series of **attributes**; each is a self-contained package of information that (hopefully) has meaning to both communicating parties.

Every RADIUS message has the same basic format, as shown in Figure 8.13. We will work through each field to explain its use and meaning.

The Code byte indicates the type of message:

- Access-Request: 1
- Access-Accept: 2
- Access-Reject: 3
- Access-Challenge: 11

The Identifier is an arbitrary number used to match up requests and replies, and the Length word indicates the total number of bytes in the message. The Authenticator is much more interesting because is has a bearing on security. The Authenticator is 16 bytes (128 bits) long and its use depends on the type of message:

In the Access-Request message, the authenticator contains a 16-byte nonce value. A nonce is a number whose value is never used twice in two different requests. In RADIUS, the nonce is used for two purposes. First, if the Access-Request message is sending a password value in an attribute, the password value is

Figure 8.13 Basic Format of RADIUS Message

encrypted using a combination of a secret key and this nonce. Second, reply messages use the nonce value in deriving an integrity check value, as described in the next paragraph.

One of three messages—Access-Accept, Access-Reject, and Access-Challenge—is sent in response to an Access-Request message. It is important to check that the reply came from the legitimate RADIUS server and has not been modified in transit. To accomplish this, an integrity check value (16 bytes) is computed and inserted into the Authenticator field of the reply.

The NAS and the RADIUS server share a secret key between them. To create the check value, the RADIUS server combines the entire reply message with the secret key. Before the computation, it inserts the nonce from the request message into the Authenticator position of the reply message and when the integrity check value has been computed, it overwrites the nonce to form a new Authenticator value. It is not practical to forge a reply that will match the request message without knowing the secret key and the use of the nonce reduces[11] the opportunity to replay an old message.

Attributes

The useful information carried in RADIUS messages is contained within attributes. Each message can carry one or more attributes and each is a self-contained package of information. It will come as no surprise that the ability to extend RADIUS depends on the ability to define and support new attributes. For a RADIUS server to be useful to you, it must support the attributes you need in your application (and the services accessed through the attributes). This is where an industry definition like WPA is useful because you can simply ask the vendor if the server conforms to the requirements of WPA. Because WPA has been designed around some common existing attributes (albeit proprietary extensions), this should not be a problem in practice, providing your RADIUS server has the required support.

Each attribute has the same format:

- A 1-byte Type field to identify the attribute
- A 1-byte Length field that indicates the number of bytes in the whole attribute
- Attribute specific data (if any)

11. We say "reduces" rather than "eliminates" because the creation of the nonce is implementation dependent and cannot be guaranteed to be unique in the true meaning of the word "nonce."

Table 8.1 Examples of RADIUS Attributes

Attribute Type Value	Name	Description
1	User-Name	The identification name or user name of the user.
2	User-Password	Contains the login password. The password data is encrypted using a shared secret and the nonce value from the Authenticator field of the Access-Request.
3	CHAP-Password	During CHAP, the user encrypts the challenge and returns a value. The value is forwarded from the NAS to the RADIUS server in this attribute.
4	NAS-IP-Address	The IP address of the NAS to which the RADIUS server should respond.
18	Reply Message	This sends text that can be displayed to the user to indicate some event or needed action.
26	Vendor-Specific	This attribute allows vendors to implement and communicate special features relevant only to their equipment. Interestingly, if they choose to make their vendor-specific attributes public, other vendors can support the features, forming a sort of nested standards process. Microsoft has done this for MS-CHAP.

There are many possible attributes. Some of the more common ones are listed in Table 8.1.

EAP over RADIUS

Because EAP was designed to extend authentication via dial-in modems, and given that so many modem pools use RADIUS, a method was needed to carry EAP over RADIUS. Extensions to RADIUS that accomplish this are described in RFC2869. These extensions are relevant to Wi-Fi LAN because WPA and RSN also use EAP. Several RADIUS extensions are defined in RFC2869. RFC2869 also has some updated procedures for sending accounting information, and it describes how to support Apple Computer's ARAP for dial-in support of Apple machines. We focus only on the section dealing with EAP over RADIUS.

In the early RADIUS standard, only two messages were available for sending authentication information between parties: Access-Request to send data from the NAS to the RADIUS server, and Access-Challenge to send data from the RADIUS server to the NAS. As the name suggests, Access-Challenge has a particular purpose similar to the challenge used in CHAP. However, RFC2869 uses this

message in a more general way to pass information back from the RADIUS server. Thus EAP messages are sent to the authentication messages inside an Access-Request message and responses are returned inside an Access-Challenge message.

The EAP message itself is sent inside one or more special attributes that have a type value of 79. All the usual EAP messages can be sent. There are a few rules to help existing RADIUS implementations map the requests to the existing conventions. For example, the identity of the dial-in user is usually sent in an EAP-Response/Identity message. This message is forwarded to the RADIUS server in an EAP attribute, but the identity information should also be copied into a User-Name attribute (type 1) and included so that RADIUS servers, including older versions, can still understand and maybe forward the message to the right place.

Recall that EAPOL includes a message called EAPOL-Start designed to kick the authenticator into action when a new device arrives and wants to get connected. RFC2869 defines a similar message called EAP-Start, which is an EAP attribute with no data. The attribute is just two bytes—a type field of 79 indicating the EAP-Message attribute and a length byte of value 2. This can be used by the NAS to get the RADIUS server started, as shown in Figure 8.14.

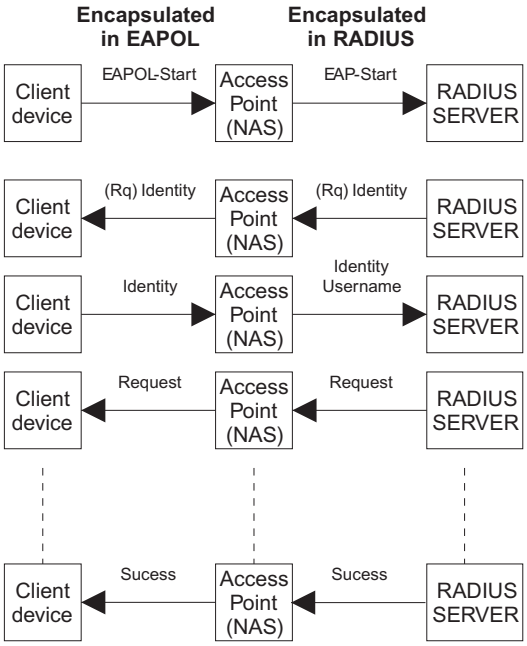

Figure 8.14 Authentication Exchange Using EAP over RADIUS

In Figure 8.14 we have shown the access point in place of the dial-up NAS, although the principles are just the same. The access point also contains an IEEE 802.1X authenticator, which talks EAP to the new client (supplicant). EAP messages that the IEEE 802.1X authenticator wants to pass back to the authentication server are packaged in RADIUS and sent to the RADIUS server. Let's step through the sequence of events.

First the new device sends an EAPOL-Start to the access point authenticator. If the access point knows that the RADIUS server supports EAP, it can go ahead and issue the EAP-Request/Identity message to the client device and send the response to the server directly. If, however, it is unsure about the server's capability, it can ask the RADIUS server to initiate the EAP exchange by sending the RADIUS server an EAP-Start message in an Access-Request message. If the server doesn't support EAP, it replays with a reject message (this is not a good idea for every exchange because the RADIUS server could be deluged with messages). If the server is EAP enabled, it sends the EAP-Request/Identity message in a RADIUS Access-Challenge message. Figure 8.14 provides an example in which the authentication method is TLS. At the end of the exchange, an EAP-Success or EAP-Fail signifies the result.

Use of RADIUS in WPA and RSN

As shown in Figure 8.14, the way RADIUS and EAP over RADIUS work fits very well with Wi-Fi WPA/RSN architecture. However, there is one important difference between the Wi-Fi and the dial-up case: For dial-up, the concern is only initial authentication, whereas WPA/RSN is concerned with establishing a lasting security context. In the dial-up case, it is only necessary to determine whether the user should be admitted to the system. Because of the nature of phone lines, an attacker is unlikely to hijack a dial-in modem once it has connected (although such an approach is theoretically possible). Therefore, once authentication is complete, there is a tendency for the NAS to sit back and assume a good guy is connected. However, as we have seen, with Wi-Fi LAN it is trivially easy to hijack an established connection just by stealing a legitimate MAC address.

Protection against session hijacking is provided by per-packet authentication and integrity protection. To provide this protection, the authentication server must pass a secret master key down to the access point. This process of generating and passing the keys is covered in great detail in Chapter 10. Earlier RADIUS servers based on RFC2865–2869 did not provide the ability to send keys from the authentication server to the NAS. The RFC assumes that the password is sent the other way for validation. However, one vendor, Microsoft, has solved this problem for another security protocol. Microsoft helped create an RFC covering their proprietary extensions to RADIUS (RFC2548: Microsoft Vendor-Specific

RADIUS Attributes). These extensions contain an attribute called MS-MPPE-Recv-Key, which is specifically intended to deliver key information to the NAS. In fact the description in the RFC says:

> The MS-MPPE-Recv-Key Attribute contains a session key for use by the Microsoft Point-to-Point Encryption Protocol (MPPE). As the name implies, this key is intended for encrypting packets received by the NAS from the remote host. This attribute is only included in Access-Accept packets.

In the IEEE 802.11i context, "MPPE" becomes "WPA" or "RSN," "NAS" becomes "access point," and "remote host" becomes "mobile device." At Microsoft's suggestion, this attribute was adopted into WPA as the recommended way to pass the master key information from the RADIUS server to the access point. It almost goes without saying that this attribute supports (and requires) encryption of the key material prior to transmission and therefore provides a more secure key delivery mechanism. Whether this attribute will make it into 802.11i is another story. The National Institute of Standards and Technology (NIST) has requested that it be deprecated in favor of a standard attribute using a key wrap algorithm.

Now we have all the pieces for using WPA/RSN in conjunction with RADIUS. The requirements are that the access point should support RADIUS, including the extensions for EAP and at least the Microsoft key delivery attribute. Also, the RADIUS server must not only support these protocols but must also understand that it is required to send the pairwise master key (PMK) to the access point (see Chapter 10). It is not mandatory under RSN to use RADIUS, although it is under WPA. Therefore, it is likely that this approach will become popular and that the RADIUS server vendors will ensure that their software provides support.

Summary

This chapter begins with a basic definition of access control. On the surface, the process of establishing the identity of the caller, checking for authorization, and opening or closing the gate is extremely simple. So simple, in fact, that the qualification requirements for a nightclub's doorman tend to be more concerned with physical mass than cranial capacity. We have seen how the three-party model of caller, security guard, and authorizer has been adopted first for dial-up modem authentication, second for LAN access authentication using IEEE 802.1X, and finally for wireless LAN authorization using IEEE 802.11 and IEEE 802.1X.

This chapter also reviewed how the messages between the three controlling parties are carefully defined using the protocols EAP and RADIUS. We observed

that wireless LAN places an additional burden on the process because it is so vulnerable to session hijack. In the case of WPA and RSN, it is necessary to establish a set of secret keys between the access point and the mobile device to protect against hijack. It this way, the authorization obtained during the access control procedure becomes like an access pass that can be used over and over with each packet of data sent.

The establishment of the secret session keys and their binding to the access control procedure has been one of the challenges of developing new security protocols (see Chapter 10). In Chapter 9, we look at the upper-level authentication protocols that ensure beyond doubt that the entities that you intend to authorize really are who they say they are.