chapter 11

Securing TCP/IP

"Better to be despised for too anxious apprehensions than ruined by too confident a security."

—Edmund Burke

In this chapter, you will learn how to

- **Discuss the standard methods for securing TCP/IP networks**
- **Compare TCP/IP security standards**
- **Implement secure TCP/IP applications**

If you really want to get into the minds of the folks who invented TCP/IP, you merely have to look at TCP/IP from the aspect of security. While today the first creators of TCP/IP are now lauded professors and leaders in our field, at the time they were for the most part pretty much just a bunch of hippies who never seriously considered the idea that evil people would get on their beloved networks and do naughty things. No part of TCP/IP has any real security. Oh sure, you can put user names and passwords on FTP, Telnet, and the like; but everything else is wide open. Those hippies must have thought that was the intent of the Internet, *openness*.

Sadly, today's world shows a totally different perspective. Every device with a public IP address on the Internet is constantly bombarded with malicious code, trying to gain some level of access to our precious data. Even data moving between two hosts is relatively easily intercepted and read. Bad guys are making millions stealing our data in one of a thousand different ways, and TCP/IP in its original form is all but powerless to stop them.

This chapter takes you on a tour of the many ways smart people have improved TCP/IP to protect our data from those who wish to do evil things to it. It's an interesting story of good intentions, knee-jerk reactions, dead ends, and failed attempts that luckily ends with a promise of easy-to-use protocols that protect our data.

This chapter examines the ways to make TCP/IP data and networks secure. I'll first give you a look at concepts of security, then turn to specific standards and protocols used to implement security. The chapter wraps with a discussion on secure TCP/IP applications and their methods.

Test Specific

Making TCP/IP Secure

I break down TCP/IP security into four areas: encryption, nonrepudiation, authentication, and authorization. **Encryption** means to scramble, to mix up, to change the data in such a way that bad guys can't read the data. Of course, this scrambled-up data must also be descrambled by the person receiving the data

Nonrepudiation is the process that guarantees that the data is as originally sent and that it came from the source you think it should have come from. Nonrepudiation is designed to cover situations in which someone intercepts your data on-the-fly and makes changes, or someone pretends to be someone they are not.

Authentication means to verify that whoever accesses the data is the person you want accessing that data. The most classic form of authentication is the user name and password combination, but there are plenty more ways to authenticate.

Authorization defines what a person accessing the data can do with that data. Different operating systems provide different schemes for authorization, but the classic scheme for Windows is to assign permissions to a user account. An administrator, for example, can do a lot more after being authenticated than a limited user can do.

Encryption, nonrepudiation, authentication, and authorization may be separate issues, but in the real world of TCP/IP security they overlap a lot. If you send a user name and password over the Internet, wouldn't it be a good idea to encrypt the user name and password so others can't read it? Equally, if you send someone a "secret decoder ring" over the Internet so he or she can unscramble the encryption, wouldn't it be a good idea for the recipient to know that the decoder ring actually came from you? In TCP/IP security, you have complete protocols that combine encryption, (sometimes) nonrepudiation, authentication, and authorization to create complete security solutions for one TCP/IP application or another.

Encryption

All data on your network is nothing more than ones and zeroes. Identifying what type of data the strings of ones and zeroes in a packet represent usually is easy. A packet of data on the network always comes with a port number, for example, so a bad guy knows what type of data he's reading.

All data starts as **plaintext**, a somewhat misleading term that simply means the data is in an easily read or viewed industry-wide standard format. Plaintext, often also referred to as **cleartext**, implies that all data starts off as text—untrue! Data often is text, but it also might be a binary file such as a photograph or an executable program. Regardless of the type of data, it all starts as plaintext. I'll use the image in Figure 11.1 as a universal figure for a piece of plaintext.

• Figure 11.1 Plaintext

If you want to take some data and make figuring out what it means difficult for other people, you need a cipher. A **cipher** is a series of complex and hard to reverse mathematics—called an **algorithm**—you run on a string of ones and zeroes to make a new set of seemingly meaningless ones and zeroes. A cipher and the methods used to implement that cipher is commonly called the **complete algorithm**. (I know that's a mouthful of new terms—check the sidebar for details.)

Tech Tip

Sorting Out the Security Terms

The terms cipher, algorithm, and complete algorithm lend themselves to a lot of confusion, especially because most people in the IT industry use them interchangeably. Here's the scoop. A cipher *is a general term for a way to encrypt data. The* algorithm *is the mathematical formula that underlies the cipher. The* complete algorithm *is both the cipher and the implementation of that cipher. The problem with the terms is compounded by the lack of a third, distinct term. Most people drop the word "complete" from "complete algorithm," for example, thus the meanings of the three terms become muddied.*

Let's say we have a string of ones and zeroes that looks like this:

```
01001101010010010100101101000101
```

This string may not mean much to you, but if it was part of an HTTP packet, your Web browser would instantly know that this is Unicode—that is, numbers representing letters and other characters—and convert it into text:

```
01001101 01001001 01001011 01000101
M I K E
```

So let's create a cipher to encrypt this cleartext. All binary encryption requires some interesting binary math. You could do something really simple such as add 1 to every value (and ignore carrying the 1):

0 + 1 = 1 and 1 + 1 = 0
```
10110010101101101011010010111010
```

No big deal; that just reversed the values. Any decent hacker would see the pattern and break this code in about three seconds. Let's try something harder to break by bringing in a second value (a key) of any eight binary numbers (let's use 10101010 for this example) and doing some math to every eight binary values using this algorithm:

If cleartext is...	And key value is...	Then the result is...
0	0	0
0	1	1
1	0	1
1	1	0

This is known as a binary XOR (eXclusive OR). Line up the key against the first eight values in the cleartext:

```
10101010
01001101010010010100101101000101
11100111
```

Then do the next eight binary values:

```
10101010
010011010100100101001011101000101
1110011111100011
```

Then the next eight:

```
101010101010101010101010
010011010100100101001011101000101
111001111110001111100001
```

Then the final eight:

```
10101010101010101010101010101010
01001101010010010100101101000101
11100111111000111110000111101111
```

If you want to decrypt the data, you need to know the algorithm and the key. This is a very simple example of how to encrypt binary data. At first glance you might say this is good encryption, but there are some downsides to this XOR'ing that makes it really easy to crack. First of all, the math is simple, and a simple XOR is easy for someone to decrypt.

An XOR works with letters as well as numbers. See if you can crack the following code:

```
WKH TXLFN EURZQ IRA MXPSV RYHU WKH ODCB GRJ
```

This is a classic example of what's known as the Caesar cipher. You just take the letters of the alphabet and transpose them:

```
Real Letter: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Code letter: DEFGHIJKLMNOPQRSTUVWXYZABC
```

Caesar ciphers are very easy to crack by using word patterns, frequency analysis, or brute force. The code "WKH" shows up twice, which means it's the same word (word patterns). We also see the letters *W* and *H* show up fairly often. Certain letters of the alphabet are used more than others, so a code-breaker can use that to help decrypt the code (frequency analysis). Assuming that you know this is a Caesar cipher, a computer can quickly go through every different code possibility and determine the answer (brute force). Incredibly, even though it's not as obvious, binary code also suffers from the same problem.

In computing you need to make a cipher hard for anyone to break except the people you want to read the data. Luckily, computers do more complex algorithms very quickly (it's just math) and you can use longer keys to make the code much harder to crack.

Okay, let's take the information above and generate some more symbols to show this process. When you run cleartext through a cipher algorithm using a key, you get what's called **ciphertext** (Figure 11.2).

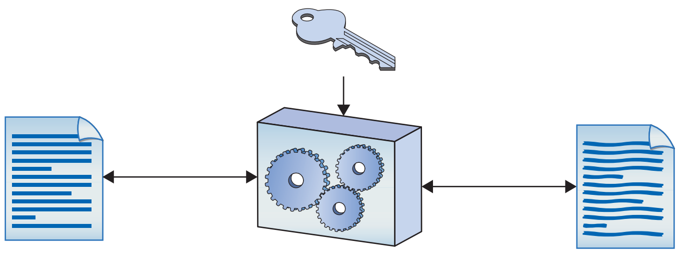


Figure 11.2 Encryption process

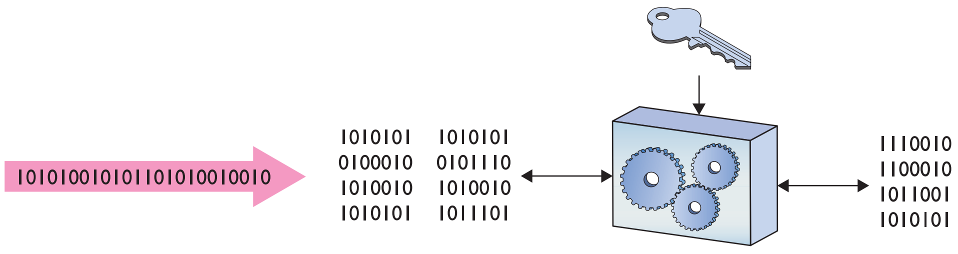
Over the years, computing people have developed hundreds of different complete algorithms for use in encrypting binary data. Of these, only a few were or still are commonly used in the TCP/IP world. The math behind all of these complete algorithms is incredibly complex, and way outside the scope of the CompTIA Network+ exam, but all of them have two items in common: a complex algorithm underlying the cipher, and a key or keys used to encrypt and decrypt the text.

Any encryption that uses the same key for both encryption and decryption is called symmetric-key encryption or a **symmetric-key algorithm**. If you want someone to decrypt what you encrypt, you have to make sure they have some tool that can handle the algorithm and you have to give them the key. This is a potential problem I will address later in this chapter.

Any encryption that uses different keys for encryption and decryption is called asymmetric-key encryption or an **asymmetric-key algorithm**. Let's look at symmetric-key encryption first, then turn to asymmetric-key encryption.

Symmetric-Key Algorithm Standards

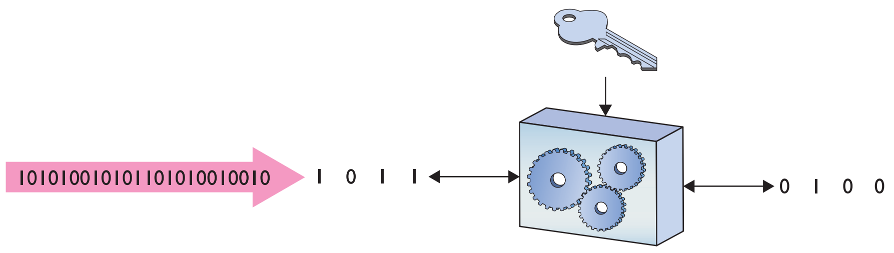
There is one difference among symmetric-key algorithms. Most algorithms we use are called **block ciphers** because they encrypt data in single "chunks" of a certain length at a time. Let's say you have a 100,000-byte Microsoft Word document you want to encrypt. One type of encryption will take 128-bit chunks and encrypt each one separately (Figure 11.3). Block ciphers work well when data comes in clearly discrete chunks. Most data crossing wired networks comes in IP packets, for example, so block ciphers are very popular with these sorts of packets.



• Figure 11.3 Block cipher

The alternative is a **stream cipher**, which takes a single bit at a time and encrypts on-the-fly (Figure 11.4). Stream ciphers are very popular whenever your data comes in long streams (such as with older wireless networks or cell phones).

The granddaddy of all TCP/IP symmetric-key algorithms is the **Data Encryption Standard (DES)**. DES was developed by the United States



• Figure 11.4 Stream cipher

government in the late 1970s and was in widespread use in a variety of TCP/IP applications. DES used a 64-bit block and a 56-bit key. Over time, the 56-bit key made DES susceptible to brute-force attacks. The computing world came up with a number of derivatives of DES to try to address this issue, with names such as 3DES, International Data Encryption Algorithm (IDEA), and Blowfish.

On the streaming side the only symmetric-key algorithm you'll probably ever see is **Rivest Cipher 4 (RC4)** stream cipher. RC4 was invented in the late 1980s by Ron Rivest, cryptographer and arguably the most famous of all inventors of TCP/IP security algorithms. RC4 is used in a number of TCP/IP applications.

Over the years improvements in computing power made both DES and RC4 vulnerable to attacks in certain circumstances. As a result almost all TCP/IP applications have moved onto **Advanced Encryption Standard (AES)**. AES is a block cipher, created in the late 1990s. It uses a 128-bit block size and 128-, 192-, or 256-bit key size. AES is incredibly secure, practically uncrackable (for now at least), and is so fast that even applications that traditionally used stream ciphers are switching to AES.

Not at all limited to just TCP/IP, you'll find AES used from file encryption to wireless networking to some Web sites. Given that AES is still somewhat new, many TCP/IP applications are still in the process of moving toward adoption.

When in doubt on a question about encryption algorithms, always pick AES. You'll be right most of the time.

Asymmetric-Key Algorithm Standards

Symmetric-key encryption has one serious weakness: anyone who gets a hold of the key can encrypt or decrypt data with it. The nature of symmetric-key encryption forces us to send the key to the other person in one way or another, making it a challenge to use symmetric-key encryption safely. As a result, there's been a strong motivation to create a methodology that allows the encryptor to send a key to the decryptor without fear of interception (Figure 11.5).

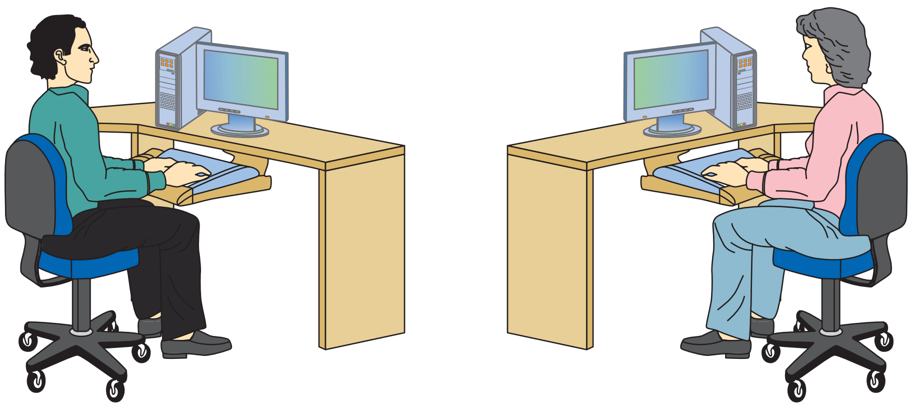
• **Figure 11.5** How do we safely deliver the key?

The answer to the problem of key sharing came in the form of using two different keys, one to encrypt and one to decrypt, an asymmetric-key algorithm. Three men in the late 1970s—Whitfield Diffie, Martin Hellman, and Ralph Merkle—introduced what became known as **public-key cryptography**, with which keys could be exchanged securely.

Ron Rivest (along with Adi Shamir and Leonard Adleman) came up with some improvements to the Diffie-Hellman method of public-key cryptography by introducing a fully functional algorithm called **Rivest Shamir Adleman (RSA)** that enabled secure digital signatures. Here's how public-key cryptography works.

Say we have two people, Mike and Melissa, who wish to send each other encrypted e-mail (Figure 11.6). SMTP doesn't have any (popular) form of encryption, so Mike and Melissa must come up with some program that

The public-key cryptography introduced by Diffie, Hellman, and Merkle became known as the *Diffie-Hellman key exchange*. Hellman, on the other hand, has insisted that if the scheme needs a name, it should be called the *Diffie-Hellman-Merkle key exchange*.



• Figure 11.6 Mike and Melissa, wanting to send encrypted e-mail messages

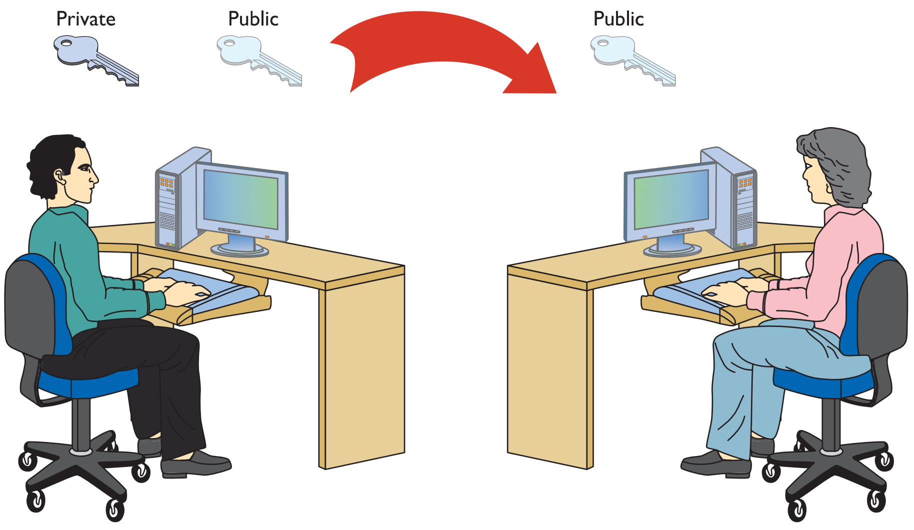
Public-key cryptography is the most popular form of e-mail encryption.

encrypts their messages. They will then send the encrypted messages as regular e-mail.

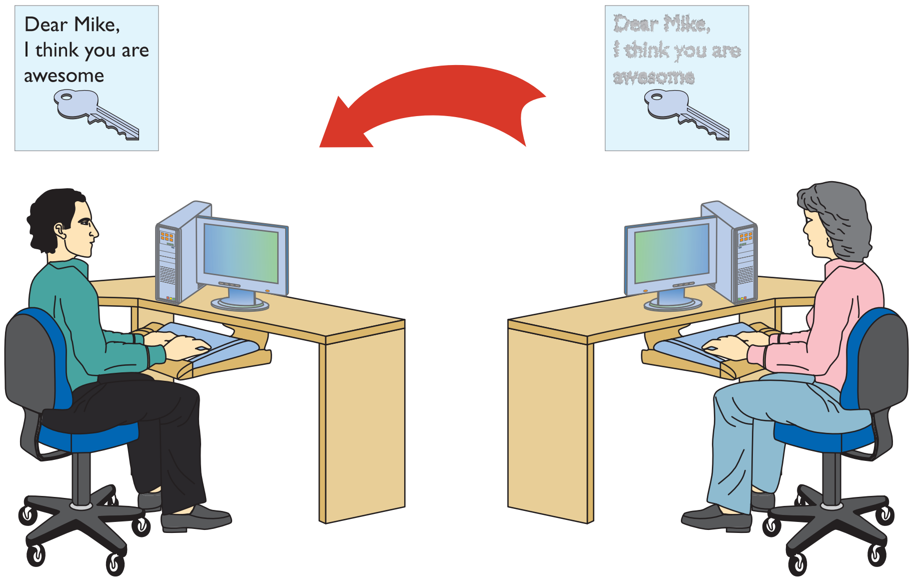
Before Melissa can send an encrypted e-mail to Mike, he first generates *two* keys. One of these keys is kept on his computer (the private key) and the other key is sent to anyone from whom he wants to receive encrypted e-mail (the public key). These two keys—called a **key pair**—are generated at the same time and are designed to work together. He sends a copy of the public key to Melissa (Figure 11.7).

A public-key cryptography algorithm works by encrypting data with a public key and then decrypting data with a private key. The public key of the key pair encrypts the data, and only the associated private key of the key pair can decrypt the data. Since Melissa has Mike's public key, Melissa can encrypt and send a message to Mike that only Mike's private key can decrypt. Mike can then decrypt the message (Figure 11.8).

If Melissa wants Mike to send encrypted e-mail to her, she must generate her own key pair and send Mike the public key. In a typical public-key



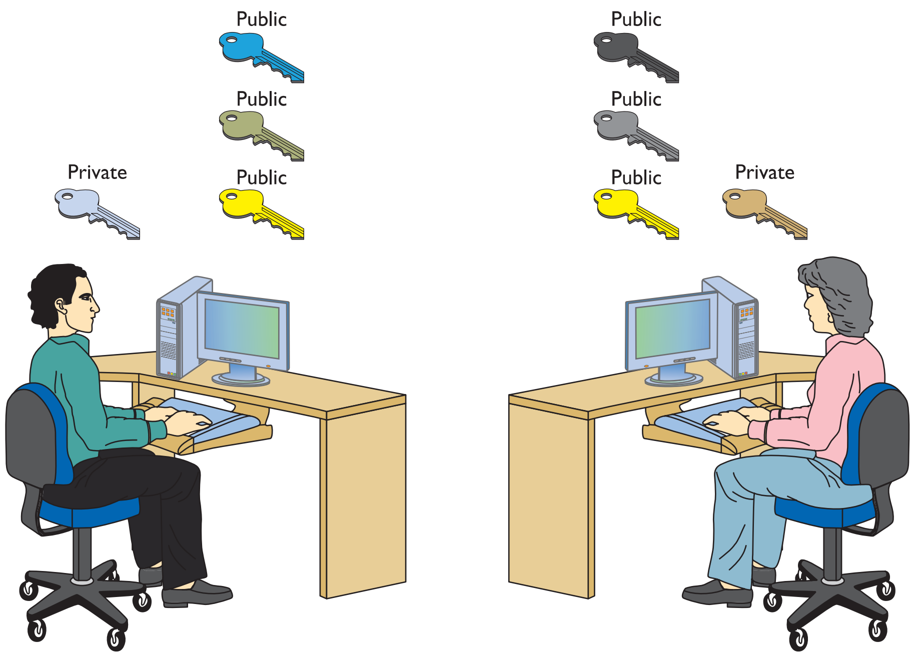
• Figure 11.7 Sending a public key



• **Figure 11.8** Decrypting a message

cryptography setup, everyone has their own private key plus a copy of the public keys for anyone with whom they wish to communicate securely (Figure 11.9).

The only problem with all these keys is the chance that someone pretending to be someone else might pass out a public key. Thus, there's a strong desire by the recipients to know who is passing out a key. This issue falls under the banner of nonrepudiation.



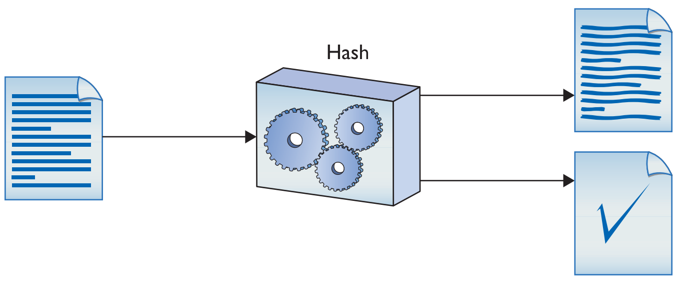
• **Figure 11.9** Lots of keys

Nonrepudiation

Within networking, nonrepudiation simply means that the receiver of information has a very high degree of confidence that the sender of a piece of information truly is who the receiver thinks it should be. Nonrepudiation takes place all over a network. Is this truly the person who sent in the user name and password to log into my Windows domain? Is this really the eBay.com Web site I'm entering my credit card number into? Did this public key really come from Mike Meyers? As a result, nonrepudiation comes in a number of forms, but most of them use a very clever little bit of mathematical magic called a hash.

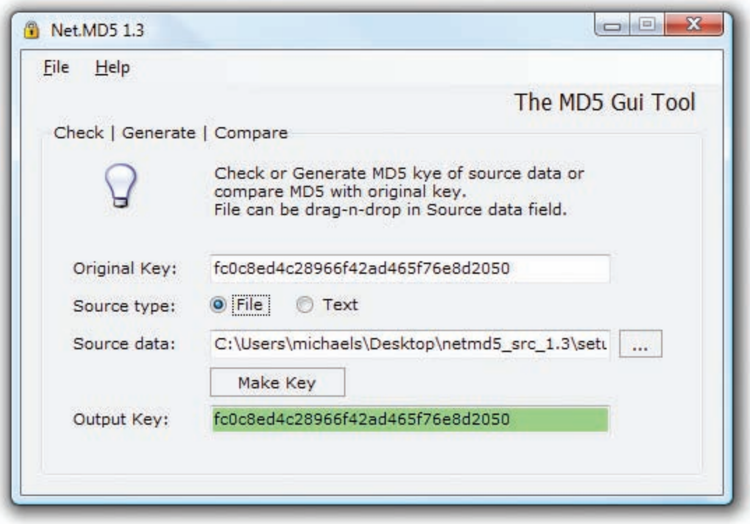
Hash

In computer security, a **hash** (or more accurately, a *cryptographic hash function*) is a mathematical function that you run on a string of binary digits of any length that results in a value of some fixed length (often called a checksum or a digest). A cryptographic hash function is a one-way function. One-way means the hash is practically irreversible. You should not be able to re-create the data even if you know the hashing algorithm and the checksum. A cryptographic hash function should also have a unique checksum for any two different input streams (Figure 11.10).



• **Figure 11.10** A hash at work

Cryptographic hash functions have a huge number of uses, but one of the most common is for files. Let's say I'm sharing a file on my Web site. I'm worried that an evil hacker might alter that file, so I run a hash on the file and supply you with both the file and the checksum. Message-Digest algorithm version 5—everybody just calls it **MD5**—is arguably the most popular hashing function for this type of work. Figure 11.11 shows an example of this, a program called Net.MD5.



Check or Generate MD5 kye of source data or compare MD5 with orignal key.
File can be drag-n-drop in Source data field.

Original Key: fc0c8ed4c28966f42ad465f76e8d2050

Source type: [✓ File] Text

Source data: C:\Users\michaels\Desktop\netmd5_src_1.3\setu

Output Key: fc0c8ed4c28966f42ad465f76e8d2050

• **Figure 11.11** File and MD5

This is only one example of how to use hashes. It's also about the only example in which you actually see hashes at work.

MD5 is a very popular cryptographic hash, but it's not the only one. The other hash you'll see from time to time is called **Secure Hash Algorithm (SHA)**. There are two versions of SHA: SHA-1 and SHA-2. Many encryption and authentication schemes also use hashes. Granted, you won't actually see the hashes as they're used, but trust me: hashes are everywhere. For example, some SMTP servers use a special form of MD5, called Challenge-Response Authentication Mechanism-Message Digest 5 (CRAM-MD5), as a tool for server authentication. (See the discussion of CHAP later in this chapter for details on how challenge-response works.) Now that you understand hashes, let's return to public-key cryptography and how a magical little something called digital signatures makes public-key cryptography even more secure.

Try This!

Doing the MD5 Thang!

Net.MD5 is a Windows program. Every operating system has lots of MD5 digest creators and checkers. If you use Linux try the popular MD5Sum utility. The following instructions are for Net.MD5:

1. Download the program from the Web site http://sourceforge.net/project/platformdownload.php?group_id=190760 and install it.
2. Download the setup_netmd5.exe.md5 file and open it in Notepad to see the MD5 digest. Copy it to the clipboard.
3. Start the Net.MD5 program.
4. Next to the **Source Data** field, browse to the Download_setup_netmd5.exe file and click **OK**.
5. Paste in the MD5 digest under the **Original Key** field.
6. Click the **Make Key** button.

Are the MD5 digests the same? Then you know you have a legit copy of Net.MD5!

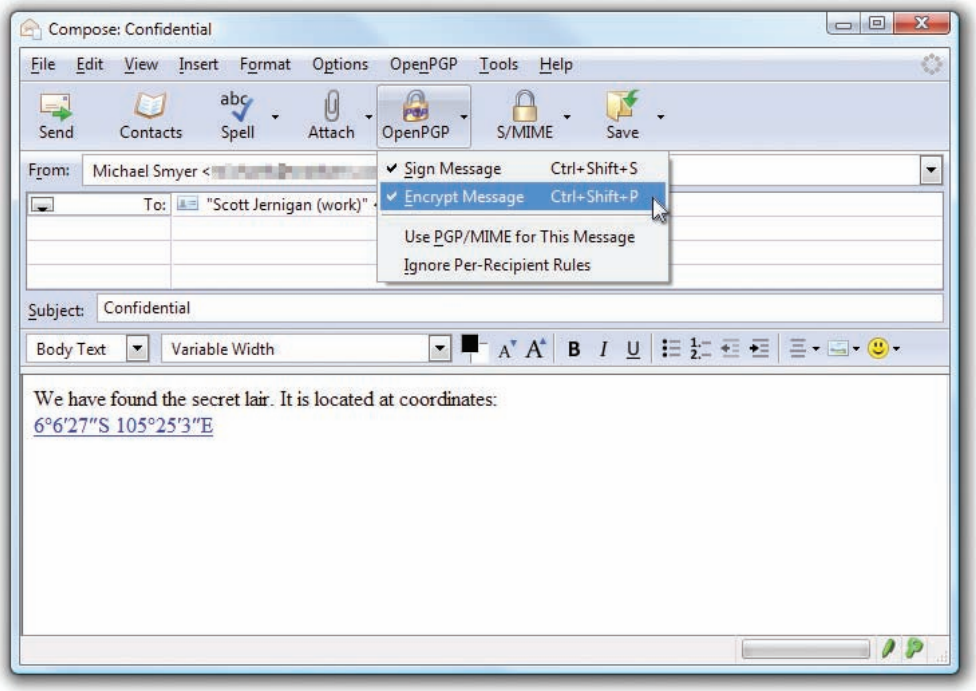
Look for CRAM-MD5 to show up on the CompTIA Network+ exam as a tool for server authentication.

Digital Signatures

As mentioned earlier, public-key cryptography suffers from the risk that you might be getting a message or a public key from someone who isn't who they say they are. To avoid this problem we add a digital signature. A **digital signature** is another string of ones and zeroes that can only be generated by the sender, usually by doing something mathematically complex (part of the algorithms always includes some hashing) to the message and the private key. The person with the matching public key does something to the digital signature using the public key to verify it came from the intended sender. Digital signatures are very popular with e-mail users. Figure 11.12 shows an e-mail message being both encrypted and digitally signed in Mozilla Thunderbird using a special Thunderbird add-on called OpenPGP. You'll read more about the PGP family of authentication/encryption tools later in this chapter.

PKI

Digital signatures are great, but what happens when you want to do business with someone you do not know? Before you enter a credit card number to buy that new USB3 Blu-ray Disc player, wouldn't you like to know that the Web site you are doing business with truly is eBay? To address that need the industry came up with the idea of certificates. A **certificate** is a standardized type of digital signature that usually includes the digital signature of a third party, a person or a company that guarantees that who is passing out this certificate truly is who they say they are. As you might imagine, certificates are


Compose: Confidential
File Edit View Insert Format Options OpenPGP Tools Help
Send Contacts Spell Attach OpenPGP S/MIME Save
Sign Message Ctrl+Shift+S
Encrypt Message Ctrl+Shift+P
Use PGP/MIME for This Message
Ignore Per-Recipient Rules
Body Text
Variable Width

From: Michael Smyer <[redacted]

To: "Scott Jernigan (work)"

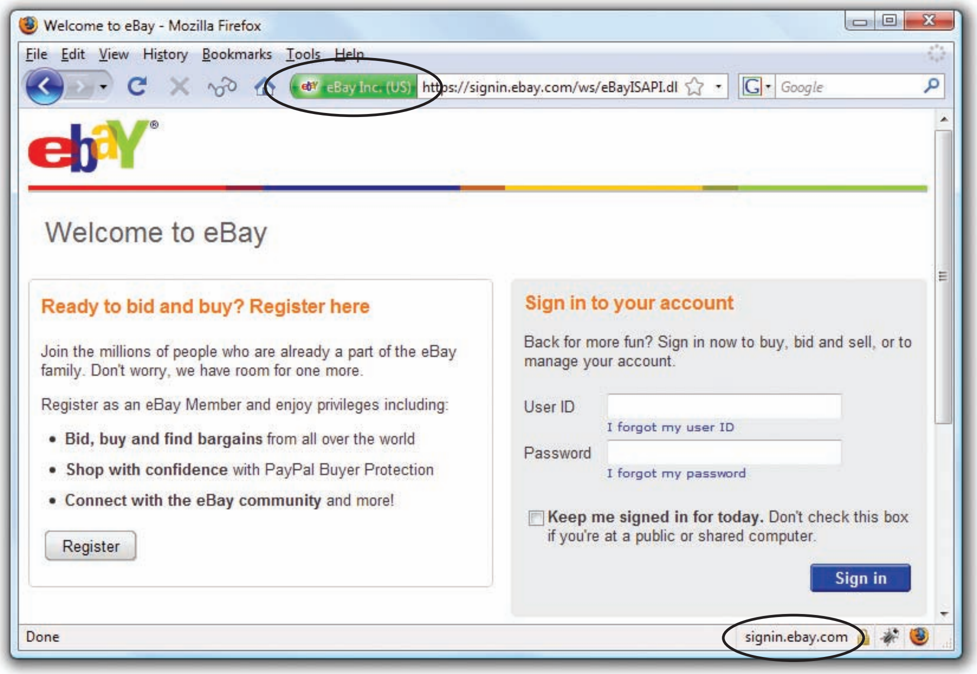
Subject: Confidential

We have found the secret lair. It is located at coordinates:
6°6′27″S 105°25′3″E

• **Figure 11.12** Digitally signed

If you see https:// or a small lock icon, you are most likely on a secure Web site.

incredibly common with secure Web pages. When you go to eBay to sign in, your browser redirects to a secure Web page. These are easy to identify by the lock icon at the bottom of the screen or in the address bar (Figure 11.13) or the https:// used (instead of http://) in the address bar.


Welcome to eBay - Mozilla Firefox
File Edit View History Bookmarks Tools Help
eBay Inc. (US)
https://signin.ebay.com/ws/eBayISAPI.dll
Google
Register
Sign in
Done
signin.ebay.com

Welcome to eBay

Ready to bid and buy? Register here

Join the millions of people who are already a part of the eBay family. Don't worry, we have room for one more.

Register as an eBay Member and enjoy privileges including:

- **Bid, buy and find bargains** from all over the world
- **Shop with confidence** with PayPal Buyer Protection
- **Connect with the eBay community** and more!

Sign in to your account

Back for more fun? Sign in now to buy, bid and sell, or to manage your account.

User ID
I forgot my user ID

Password
I forgot my password

Keep me signed in for today. Don't check this box if you're at a public or shared computer.

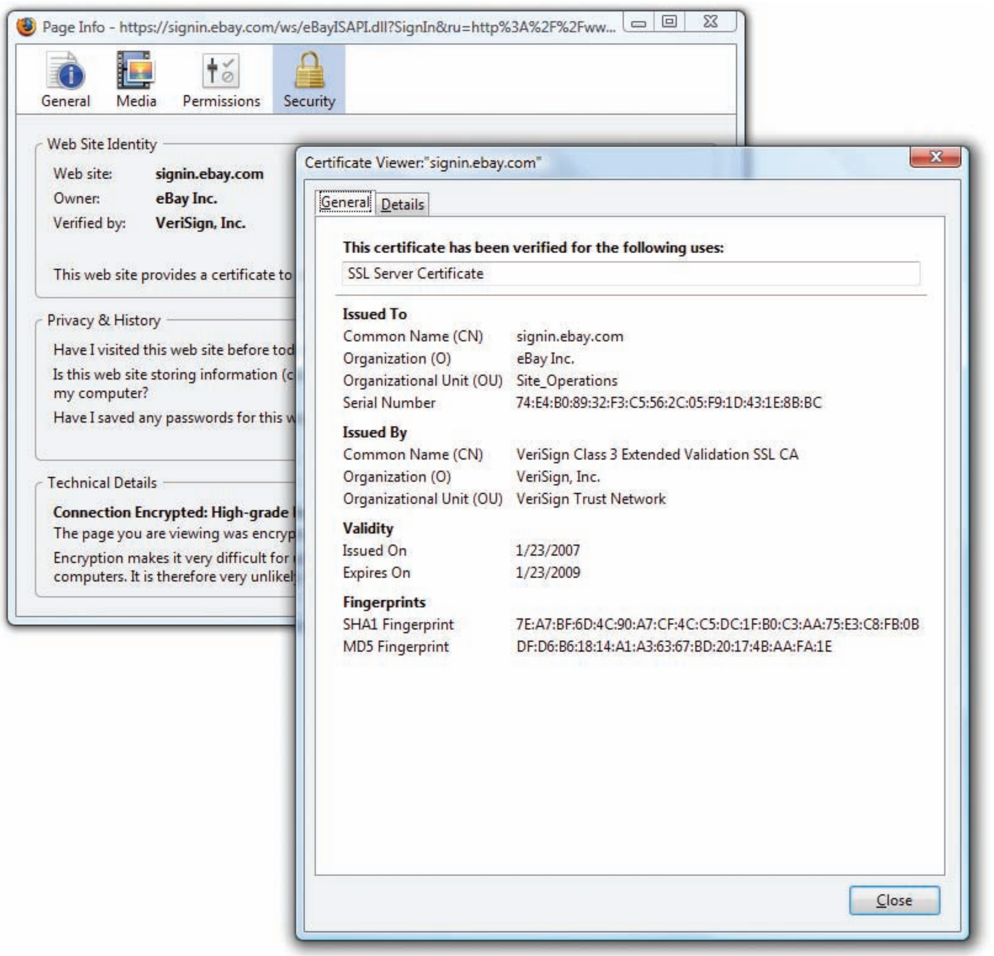
• **Figure 11.13** Secure Web page

In the background, several items take place (all before the secure Web page loads). First, the Web server automatically sends down a copy of its certificate. Built into that certificate is the Web server's digital signature and a signature from the third party who guarantees this is really eBay. Go to your national version of eBay (I'm in the United States, so I'll use eBay.com) and click **Sign In** (you don't even need an eBay account to do this). Now look at the certificate for the current session. Depending on the Web browser you use, there are different ways to see it. Try clicking the little lock icon at the bottom of the page or in the address bar as this usually works. Figure 11.14 shows the certificate for this session.

So there's a company called VeriSign that issued this certificate. That's great, but how does your computer check all this? Simple, VeriSign is a certificate authority. Every Web browser keeps a list of certificate authority certificates that it checks against when it receives a digital certificate. Figure 11.15 shows the certificate authority certificates stored on my system.

When someone wants to create a secure Web site, he or she buys a certificate signed by a certificate authority, such as VeriSign (the biggest player in the market and the one I'll use for this example). VeriSign acts as the root,

It is very difficult to become a root certificate authority with enough respect to have Web browsers install your certificate!



Page Info - https://signin.ebay.com/ws/eBayISAPI.dll?SignIn&ru=http%3A%2F%2Fww...
General
Media
Permissions
Security
Web Site Identity
Privacy & History
Technical Details
Certificate Viewer:"signin.ebay.com"
General
Details
Close

Web site: signin.ebay.com
Owner: eBay Inc.
Verified by: VeriSign, Inc.

This web site provides a certificate to

Have I visited this web site before tod
Is this web site storing information (c
my computer?
Have I saved any passwords for this w

Connection Encrypted: High-grade
The page you are viewing was encryp
Encryption makes it very difficult for
computers. It is therefore very unlikel

This certificate has been verified for the following uses:
SSL Server Certificate

Issued To

Common Name (CN)	signin.ebay.com
Organization (O)	eBay Inc.
Organizational Unit (OU)	Site_Operations
Serial Number	74:E4:B0:89:32:F3:C5:56:2C:05:F9:1D:43:1E:8B:BC

Issued By

Common Name (CN)	VeriSign Class 3 Extended Validation SSL CA
Organization (O)	VeriSign, Inc.
Organizational Unit (OU)	VeriSign Trust Network

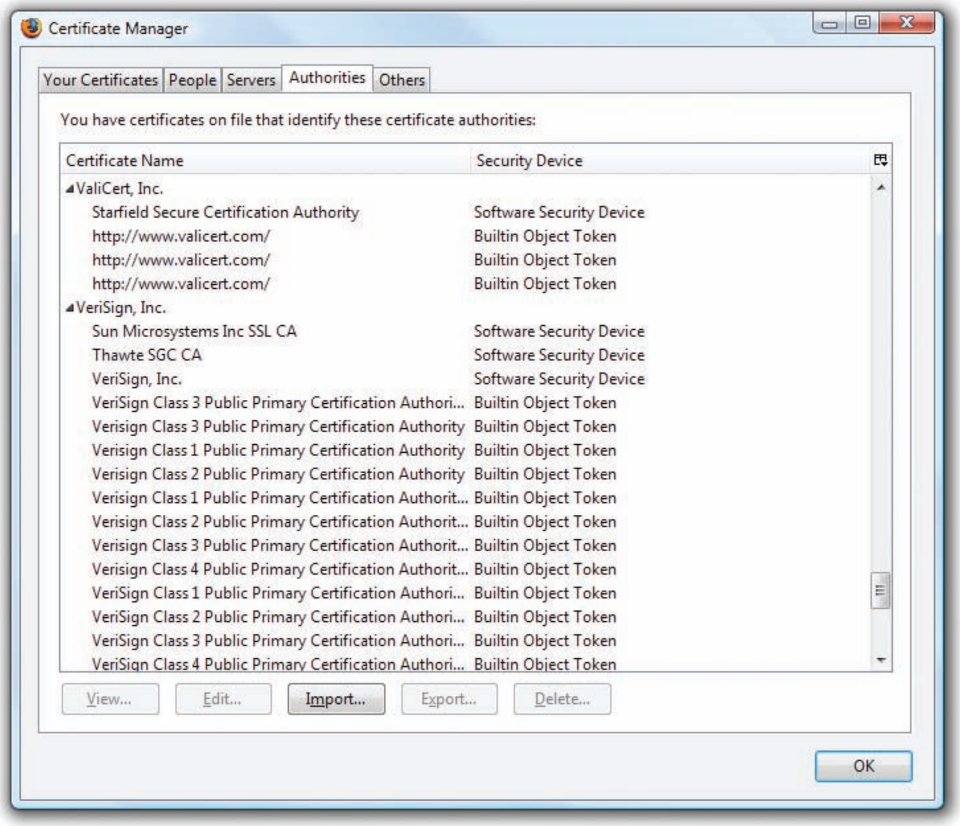
Validity

Issued On	1/23/2007
Expires On	1/23/2009

Fingerprints

SHA1 Fingerprint	7E:A7:BF:6D:4C:90:A7:CF:4C:C5:DC:1F:B0:C3:AA:75:E3:C8:FB:0B
MD5 Fingerprint	DF:D6:B6:18:14:A1:A3:63:67:BD:20:17:4B:AA:FA:1E

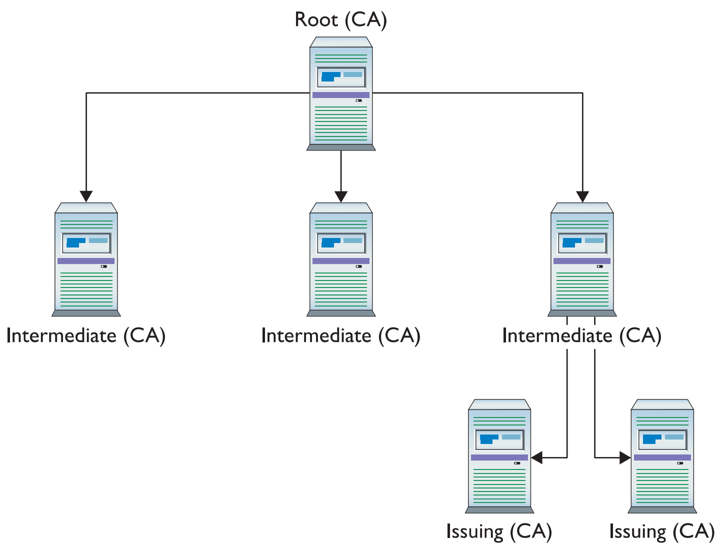
• **Figure 11.14** eBay sign-in certificate



You have certificates on file that identify these certificate authorities:

Certificate Name	Security Device
◢ValiCert, Inc.	
Starfield Secure Certification Authority	Software Security Device
http://www.valicert.com/	Builtin Object Token
http://www.valicert.com/	Builtin Object Token
http://www.valicert.com/	Builtin Object Token
◢VeriSign, Inc.	
Sun Microsystems Inc SSL CA	Software Security Device
Thawte SGC CA	Software Security Device
VeriSign, Inc.	Software Security Device
VeriSign Class 3 Public Primary Certification Authori...	Builtin Object Token
Verisign Class 3 Public Primary Certification Authority	Builtin Object Token
Verisign Class 1 Public Primary Certification Authority	Builtin Object Token
Verisign Class 2 Public Primary Certification Authority	Builtin Object Token
Verisign Class 1 Public Primary Certification Authorit...	Builtin Object Token
Verisign Class 2 Public Primary Certification Authorit...	Builtin Object Token
Verisign Class 3 Public Primary Certification Authorit...	Builtin Object Token
Verisign Class 4 Public Primary Certification Authorit...	Builtin Object Token
VeriSign Class 1 Public Primary Certification Authori...	Builtin Object Token
VeriSign Class 2 Public Primary Certification Authori...	Builtin Object Token
VeriSign Class 3 Public Primary Certification Authori...	Builtin Object Token
VeriSign Class 4 Public Primary Certification Authori...	Builtin Object Token

• Figure 11.15 Certificate authority certificates on system



• Figure 11.16 VeriSign's PKI tree

and the new Web site's certificate contains VeriSign's signature. For more advanced situations, VeriSign includes an intermediate certificate authority between VeriSign's root certificate authority and the user's certificate. This creates a tree of certificate authorization, with the root authorities at the top and issued certificates at the bottom. You can also have intermediate authorities, although these are not as heavily used. Together, this organization is called a **public-key infrastructure (PKI)** (Figure 11.16).

You don't have to use PKI to use certificates. First, you can create your own, unsigned, certificates. These are perfectly fine for lower-security situations (e-mail among friends, personal Web page, and so forth), but don't expect anyone to buy products on a Web site or send highly sensitive e-mail without a signed certificate from a well-known certificate authority like VeriSign, thawte, or GoDaddy.

Last, many certificate providers, primarily for e-mail, provide a Web-of-trust option. In this case, someone else who is already part of a trust group signs your certificate. There is no certificate authority, just a group of peers who trust each other. The popular Pretty Good Privacy (PGP) encryption program, among many others, uses such a trust model.

Digital certificates and asymmetric cryptography are closely linked as digital certificates are almost always used to verify the exchange of public keys. In many cases this takes place behind the scenes of e-mail, Web pages, and even in some very secure wireless networks. Though you may not see certificates in action very often, do know they are there.

Tech Tip

Get in the Game

Almost all e-mail clients support encryption—you just need to get a certificate. If you want to start playing with e-mail encryption and signing, grab a free personal e-mail certificate from thawte: www.thawte.com. You can even join thawte's Web of Trust (WOT). Instructions are on the Web site.

Authentication

You most likely have dealt with authentication at some level. Odds are good you've at least had to type in a user name and password on a Web site. Maybe your computer at work or school requires you to log onto the network. Whatever the case, the first exposure to authentication for most users is a request to enter a user name and password. A network technician should understand not only how different authentication methods control user names and passwords, but also some of the authentication standards used in today's TCP/IP networks.

Passwords offer significant security challenges. What happens after you type in a user name and password? How is this data transferred? Who or what reads this? What is the data compared to? A series of TCP/IP security standards that use combinations of user names, passwords, and sometimes certificates, all handled in a usually secure manner, address these issues, as described in the upcoming section "TCP/IP Security Standards."

Authorization

A large part of the entire networking process involves one computer requesting something from another computer. A Web client might ask for a Web page, for example, or a Common Internet File System (CIFS) client might ask a file server for access to a folder. A computer far away might ask another computer for access to a private network. Whatever the case, you should carefully assign levels of access to your resources. This is authorization. To help define how to assign levels of access, you use an access control list.

An **access control list (ACL)** is nothing more than a clearly defined list of permissions that specify what an authenticated user may perform on a shared resource. Over the years the way to assign access to resources has changed dramatically. To help you to understand these changes, the security industry likes to use the idea of *ACL access models*. There are three types of ACL access models: mandatory, discretionary, and role based.

In a **mandatory access control (MAC)** security model, every resource is assigned a label that defines its security level. If the user lacks that security level, they do not get access. MAC is used in many operating systems to define what privileges programs have to other programs stored in RAM. The MAC security model is the oldest and least common of the three.

Discretionary access control (DAC) is based on the idea that there is an owner of a resource who may at his or her discretion assign access to that resource. DAC is considered much more flexible than MAC.

Role-based access control (RBAC) is the most popular model used in file sharing. RBAC defines a user's access to a resource based on the roles the user plays in the network environment. This leads to the idea of creation of groups. A group in most networks is nothing more than a name that has clearly defined accesses to different resources. User accounts are placed into various groups. A network might have a group called "Sales" on a Web server that gives any user account that is a member of the Sales group access to a special Web page that no other groups can see.

Keep in mind that these three types of access control are models. Every TCP/IP application and operating system has its own set of rules that sometimes follow one of these models, but in many cases does not. But do make sure you understand these three models for the CompTIA Network+ exam!

TCP/IP Security Standards

Now that you have a conceptual understanding of encryption, nonrepudiation, authentication, and authorization, it's time to see how the TCP/IP folks have put it all together to create standards so that you can secure just about anything in TCP/IP networks.

TCP/IP security standards are a rather strange mess. Some are authentication standards, some are encryption standards, and some are so unique to a single application that I'm not even going to talk about them in this section and instead will wait until the "Secure TCP/IP Applications" discussion at the end of this chapter. There's a reason for all this confusion: TCP/IP was never really designed for security. As you read through this section, you'll discover that almost all of these standards either predate the whole Internet, are slapped-together standards that have some serious issues, or, in the case of the most recent standards, are designed to combine a bunch of old, confusing standards. So hang tight—it's going to be a bumpy ride!

Authentication Standards

Authentication standards are some of the oldest standards used in TCP/IP. Many are so old they predate the Internet itself. There was a time when nobody had fiber-optic, cable modem, or DSL connections to their ISPs. For the most part, if you wanted to connect to the Internet you had a choice: go to the computer center or use dial-up.

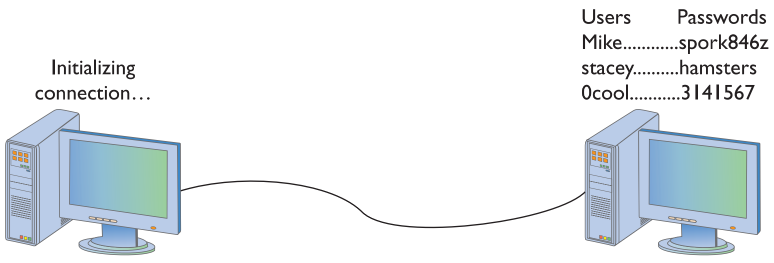
Dial-up, using for the most part telephone lines, predates the Internet, but the nerds of their day didn't want just anybody dialing into their computers. To prevent unauthorized access, they developed some excellent authentication methods that TCP/IP adopted for itself. A number of authentication methods were used back in these early days, but for the most part TCP/IP authentication started with something called the Point-to-Point Protocol.

PPP

The **Point-to-Point Protocol (PPP)** enables two point-to-point devices to connect, authenticate with a user name and password, and negotiate the network protocol the two devices will use. Today that network protocol is almost always TCP/IP.

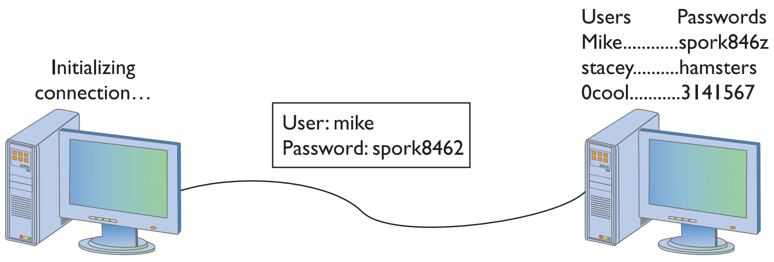
Note that point-to-point and dial-up are not Ethernet, but still can support TCP/IP. Many network technologies don't need Ethernet, such as telephone, cable modem, microwave, and wireless (plus a bunch more you won't even see until Chapter 14, "Remote Connectivity"). In fact, once you leave a LAN, most of the Internet is just a series of point-to-point connections.

PPP provided the first common method to get a server to request a user name and password. In such a point-to-point connection, the side asking for the connection is called the initiator while the other side, which has a list of user names and passwords, is called the authenticator (Figure 11.17).



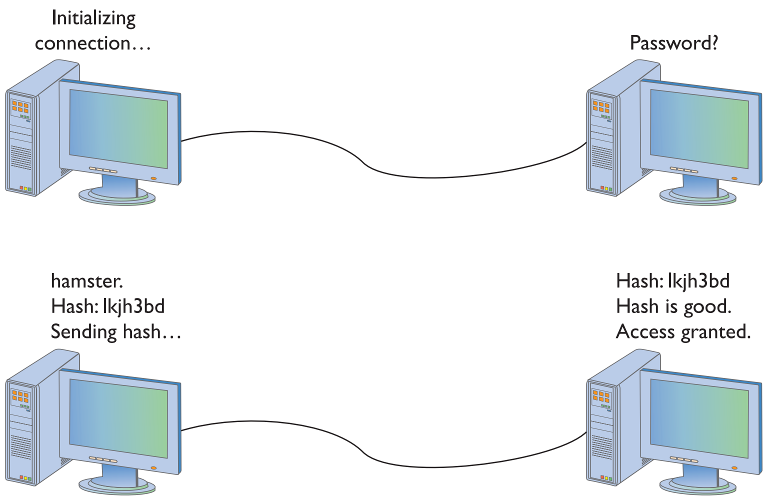
● **Figure 11.17** A point-to-point connection

PPP came with two methods to authenticate a user name and password. The original way—called **Password Authentication Protocol (PAP)**—simply transmits the user name and password over the connection in plaintext. Unfortunately, that means that anyone who could tap the connection could learn the user name and password (Figure 11.18).



● **Figure 11.18** PAP in action

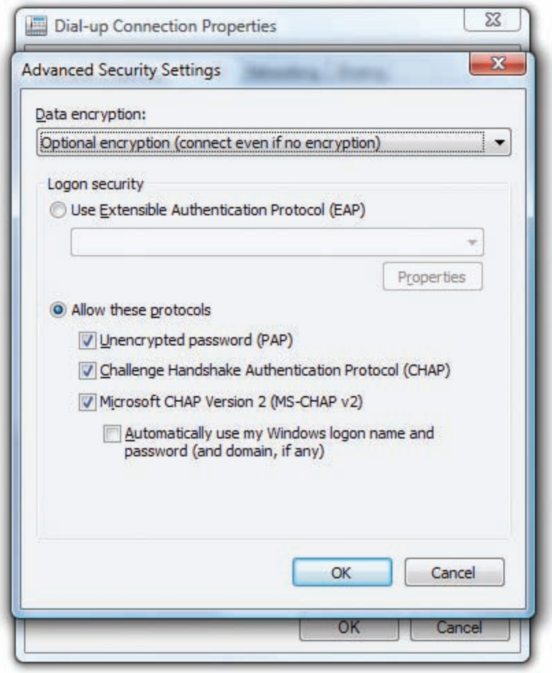
Fortunately, with PPP you also have the choice of the safer **Challenge Handshake Authentication Protocol (CHAP)** to provide a more secure authentication routine. CHAP relies on hashes based on a shared secret, usually a password that both ends of the connection know. When the initiator of the connection makes the initial connection request, the authenticator creates some form of challenge message. The initiator then makes a hash using the password and sends that to the authenticator. The authenticator in turn compares that value to its own hash calculation based on the password. If they match, the initiator is authenticated (Figure 11.19).



● **Figure 11.19** CHAP in action

CHAP works nicely because it never sends the actual password over the link. The CHAP standard leaves a number of issues undefined, however, like "If the hash doesn't match, what do I do?" The boom in dial-up connections to the Internet in the 1990s led Microsoft to invent a more detailed version of CHAP called **MS-CHAP**. The current version of MS-CHAP is called MS-CHAPv2. MS-CHAPv2 is still the most common authentication method for the few of us using dial-up connections. Dial-up is still being used out there and even the latest operating systems support it. Figure 11.20 shows the dial-up connection options for Vista.

Yes, I still have a dial-up connection account that I use when nothing else is available.


Dial-up Connection Properties

Advanced Security Settings

Data encryption:

Optional encryption (connect even if no encryption)

Logon security

Use Extensible Authentication Protocol (EAP)

Properties

[✓ Allow these protocols]

☑ Unencrypted password (PAP)

☑ Challenge Handshake Authentication Protocol (CHAP)

☑ Microsoft CHAP Version 2 (MS-CHAP v2)

☐ Automatically use my Windows logon name and password (and domain, if any)

OK Cancel

● Figure 11.20 MS-CHAP is alive and well

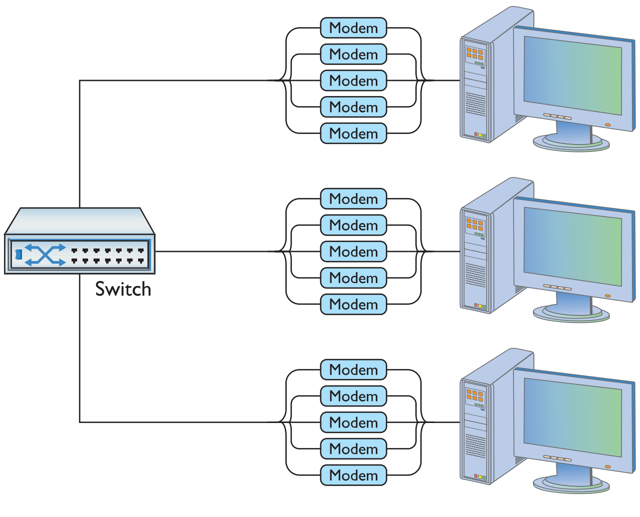
If you get a question on PAP, CHAP, and MS-CHAP on the CompTIA Network+ exam, remember that MS-CHAP offers the most security.

AAA

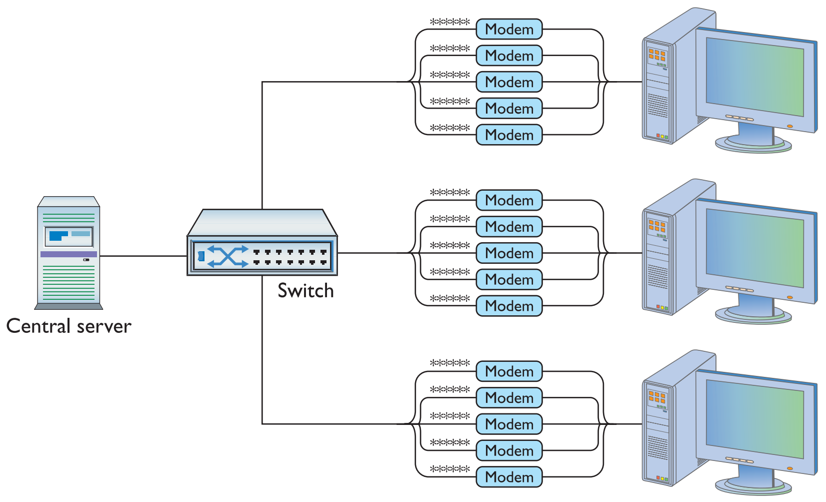
PPP does a great job of handling authentication for point-to-point connections, but it has some limitations. The biggest problem is that, in many cases, a network might have more than one point for an initiator to enter. PPP assumes that the authenticator at the endpoint has all the user name and password information, but that's not necessarily true. In traditional modem communication, for example, an Internet service provider (ISP) has a large bank of modems to support any number of users. When a user dials in, the modem bank provides the first available connection, but that means that any modem in the bank has to support any of the users. You can't put the database containing all user names and passwords on every modem (Figure 11.21).

In this case you need a central database of user names and passwords. That's simple enough, but it creates another problem—anyone accessing the network can see the passwords (Figure 11.22). PPP is good at the endpoints, but once the data gets on the network it's unencrypted.

Thus the folks overseeing central databases full of user names and passwords needed to come up with standards to follow to protect that data. They first agreed upon a little philosophy called **Authentication, Authorization, and Accounting (AAA)**. AAA is designed for the idea



● Figure 11.21 Where do you put the user names and passwords?



• **Figure 11.22** Central servers are prone to attack.

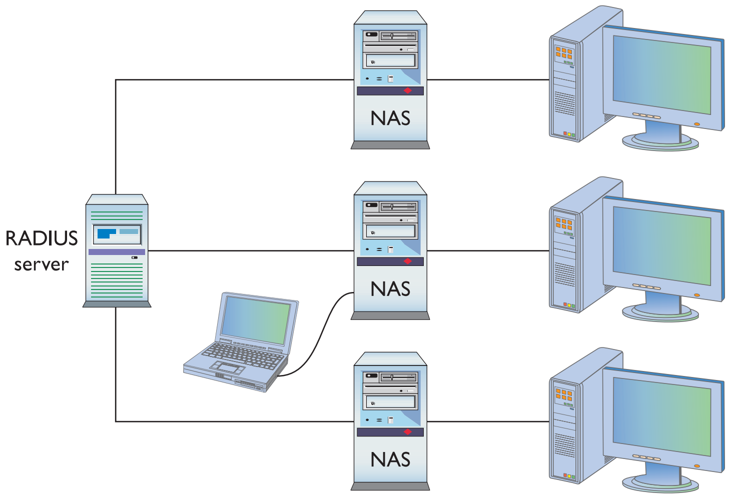
of port authentication—the concept of allowing remote users authentication to a particular point-of-entry (a port) to another network.

- **Authentication** A computer that is trying to connect to the network must present some form of credential for access to the network. This credential is most commonly a user name and password, but it might also be a security token such as a smart card, retinal scan, or digital certificate. It might even be a combination of some of these. The authentication gives the computer the right to access the network.
- **Authorization** Once authenticated, the computer determines what it can or cannot do on the network. It might only be allowed to use a certain amount of bandwidth. It might be limited to working only certain times of day or might be limited to using only a certain set of applications.
- **Accounting** The authenticating server should do some form of accounting such as recording the number of times a user logs on and logs off. It might track unsuccessful logon attempts. It may track what services or resources the client system accessed. The number of items to be accounted is massive.

Once the idea of AAA took shape, those smart Internet folks got to work developing two standards: RADIUS and TACACS+. Both standards offer authentication, authorization, and accounting.

RADIUS **Remote Authentication Dial-In User Service (RADIUS)** is the better known of the two AAA standards and, as its name implies, was created to support ISPs with hundreds if not thousands of modems in hundreds of computers to connect to a single central database. RADIUS consists of three devices: the RADIUS server that has access to a database of user names and passwords, a number of **Network Access Servers (NASs)** that control the modems, and a group of systems that dial into the network (Figure 11.23).

NAS stands for either Network Access Server or Network Attached Storage. Make sure you read the question to see which NAS it's looking for!



• Figure 11.23 RADIUS setup

To use RADIUS you need a RADIUS server. The most popular choice for Microsoft environments is **Internet Authentication Service (IAS)**. IAS comes built in with most versions of Microsoft Windows Server operating systems. For the UNIX/Linux crowd, the popular (yet in my opinion hard to set up) **FreeRADIUS** is the best choice. If you prefer a more prepackaged server, you might look at Juniper Network's Steel-Belted RADIUS—avery powerful and somewhat easy to set up option that many people feel is well worth the roughly $3000 price tag

A single RADIUS server can support multiple NASs and provide a complete PPP connection from the requesting system, through the NAS, all the way to the RADIUS server. Like any PPP connection, the RADIUS server supports PAP, CHAP, and MS-CHAP. Even if you use PAP, RADIUS hashes the password so that at no time is the user name/password exposed. Newer versions of RADIUS support even more authentication methods, as you will soon see. RADIUS performs all this authentication on either UDP ports 1812 and 1813 or UDP ports 1645 and 1646.

TACACS+ Routers and switches need administration. In a simple network, you can access the administration screen for each router and switch by entering a user name and password for each device. When a network becomes complex, with many routers and switches, logging into each device separately starts to become administratively messy. The answer is to make a single server store the ACL for all the devices in the network. To make this secure, you need to follow the AAA principles.

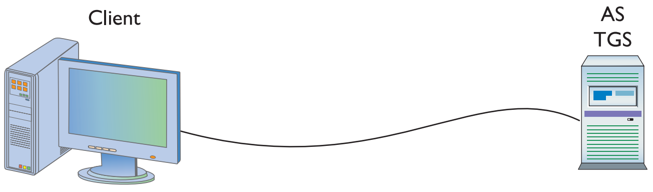
Terminal Access Controller Access Control System Plus (TACACS+) is a proprietary protocol developed by Cisco to support AAA in a network with many routers and switches. TACACS+ is very similar to RADIUS in function, but uses TCP port 49 by default and separates authorization, authentication, and accounting into different parts. TACACS+ uses PAP, CHAP, and MD5 hashes, but can also use something called Kerberos as part of the authentication scheme.

Kerberos

Up to this point almost all the authentication schemes we've discussed either are based on PPP or at least take the idea of PPP and expand upon it. Of course, every rule needs an exception and Kerberos is the exception here.

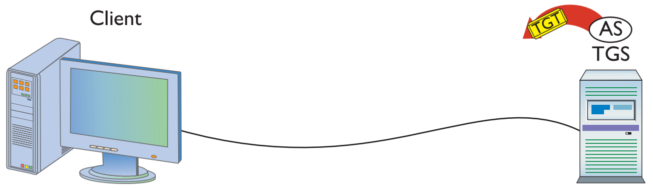
Kerberos is an authentication protocol that has no connection to PPP. Twenty years ago some Internet folks began to appreciate that TCP/IP was not secure and thus designed Kerberos. Kerberos is an authentication protocol for TCP/IP networks with many clients all connected to a single authenticating server—no point-to-point here! Kerberos works nicely in a network, so nicely that Microsoft adopted it as the authentication protocol for all Windows networks using a domain controller.

The cornerstone of Kerberos is the **Key Distribution Center (KDC)**, which has two processes: the **Authentication Server (AS)** and the Ticket-Granting Service (TGS). In Windows server environments, the KDC is installed on the domain controller (Figure 11.24).



• **Figure 11.24** Windows Kerberos setup

When your client logs onto the domain, it sends to the AS a request that includes a hash of the user name and password. The AS compares the results of that hash to its own hash (as it also stores the user name and password) and, if they match, sends a **Ticket-Granting Ticket (TGT)** and a timestamp (Figure 11.25). The ticket has a default lifespan in Windows of eight hours. The client is now authenticated, but not yet authorized.



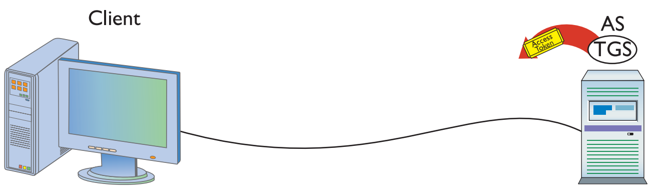
• **Figure 11.25** AS sending a TGT back to client

The client then sends the timestamped TGT to the TGS for authorization. The TGS sends a timestamped service ticket (also called a token or access token) back to the client (Figure 11.26).

This token is the key that the client uses to access any resource on the entire domain. This is where authorization takes place. The token authorizes the user to access resources without "reauthenticating." Any time the client attempts to access a folder, printer, or service anywhere in the domain, the server sharing that resource uses the token to see exactly what access the client may have to that resource.

In Windows, the security token is called a Security Identifier (SID).

Timestamping is important for Kerberos because it forces the client to request a new token every eight hours. This prevents third parties from intercepting the tokens and attempting to crack them. Kerberos tokens can be cracked, but it's doubtful this can be done in under eight hours.



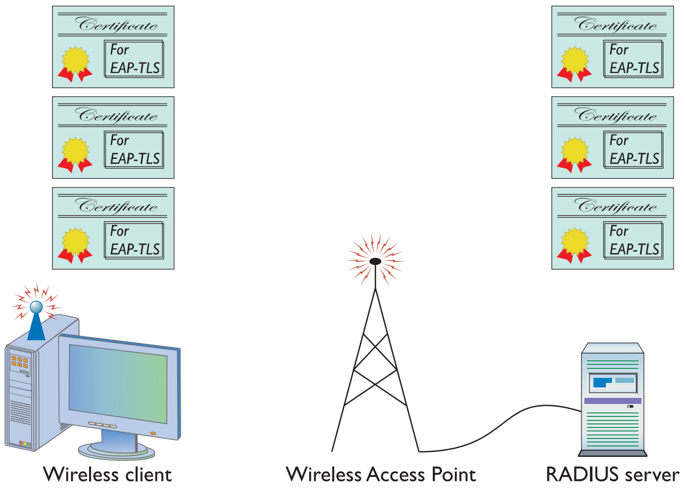
• **Figure 11.26** TGS sending token to client

Kerberos is very popular, but has some serious weaknesses. First, if the KDC goes down, no one has access. That's why Microsoft and other operating systems that use Kerberos always stress the importance of maintaining a backup KDC. In Windows it is standard practice to have at least two domain controllers. Second, timestamping requires that all the clients and servers synchronize their clocks. This is fairly easy to do in a wired network (such as a Windows domain or even a bunch of connected routers using TACACS+), but adds an extra level of challenge in dispersed networks (such as those connected across the country).

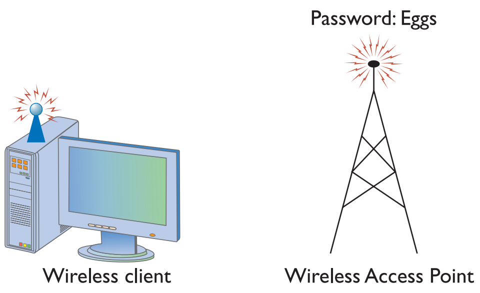
EAP

One of the great challenges to authentication is getting the two ends of the authentication process to handle the many different types of authentication options. The **Extensible Authentication Protocol (EAP)** was developed to help two devices negotiate. Despite the name, EAP is not a protocol in the classic sense, but rather it is a wrapper that EAP-compliant applications can use to accept one of many types of authentication. While EAP is a general-purpose authentication wrapper, its only substantial use is in wireless networks. (See Chapter 16, "Wireless Networking," to see where EAP is used.) EAP comes in various types, but currently only six types are in common use:

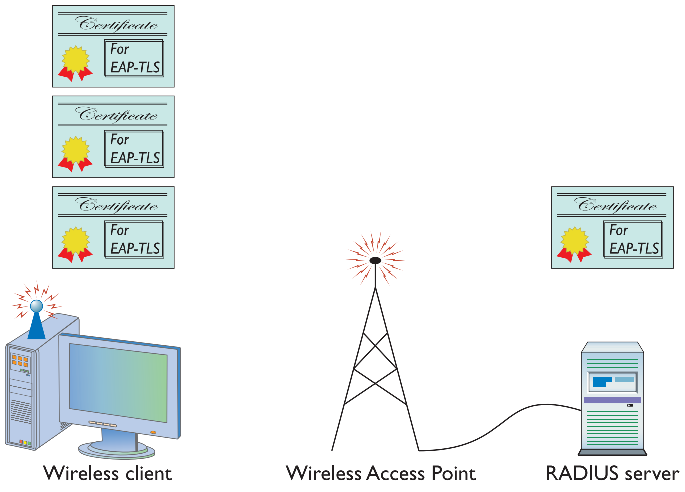
- **EAP-PSK** Easily the most popular form of authentication used in wireless networks today, EAP-PSK (Personal Shared Key) is nothing more than a shared secret code that's stored on both the Wireless Access Point (WAP) and the wireless client, encrypted using the powerful AES encryption (Figure 11.27). See Chapter 16 for the scoop on WAPs and EAP.
- **EAP-TLS** EAP with Transport Layer Security (TLS) defines the use of a RADIUS server as well as mutual authentication, requiring certificates on both the server and every client. On the client side, a smart card may be used in lieu of a certificate. EAP-TLS is very robust, but the requirement of having certificates on the client side is an administrative challenge. Even though it's a challenge, the most secure wireless networks all use EAP-TLS. EAP-TLS is only used on wireless networks, but TLS is used heavily on secure Web sites (see the section "SSL/TLS" later in this chapter). Figure 11.28 shows a typical EAP-TLS setup for a wireless network.
- **EAP-TTLS** EAP-TTLS (Tunneled TLS) is similar to EAP-TLS but only uses a single server-side certificate. EAP-TTLS is very common for more secure wireless networks (Figure 11.29).
- **EAP-MS-CHAPv2, more commonly known as PEAP** **Protected Extensible Authentication Protocol (PEAP)** uses a password function based on MS-CHAPv2 with the addition of an encrypted TLS tunnel similar to EAP-TLS.
- **EAP-MD5** This is a very simple version of EAP that uses only MD5 hashes for transfer of authentication credentials. EAP-MD5 is weak and the least used of all the versions of EAP described.
- **LEAP** **Lightweight Extensible Authentication Protocol (LEAP)** is a proprietary EAP authentication used almost exclusively by Cisco wireless products. LEAP is an interesting combination of MS-CHAP authentication between a wireless client and a RADIUS server.



• Figure 11.27 EAP-PSK in action



• Figure 11.28 EAP-TLS



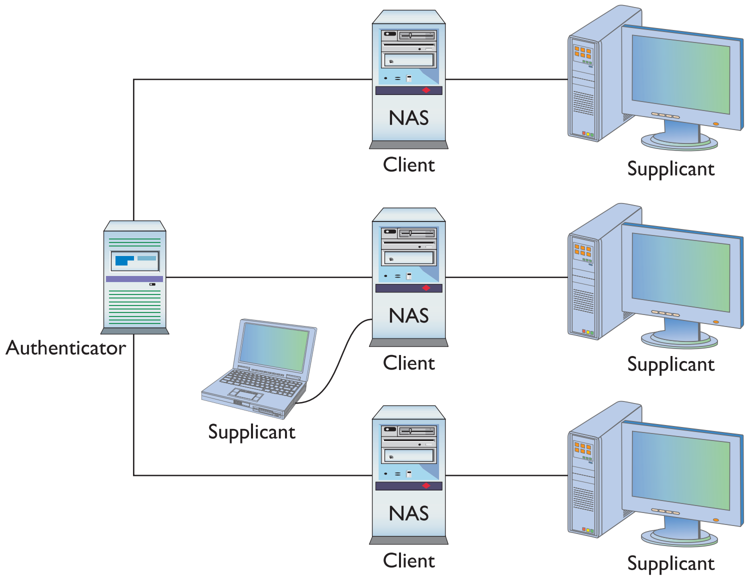
• Figure 11.29 EAP-TTLS

802.1X

When wireless networks first came out many years ago, they lacked any form of authentication. After many fits and starts the wireless community grabbed a preexisting authentication standard called 802.1X to use in their wireless networks.

802.1X is a port-authentication network access control mechanism for networks. In other words, it's a complete authentication standard designed to force devices to go through a full AAA process to get anywhere past the interface on a gateway system. Before 802.1X, a system on a wired network could always access another system's port. Granted, an attacker wouldn't be able to do much until he gave a user name/password or certificate, but he could still send packets to any computer on the network. This wasn't good, because it enabled attackers to get to the systems to try to do evil things. 802.1X prevented them from even getting in the door until they were authenticated and authorized.

The interesting part is that you already know about most of the parts of 802.1X, because the standard worked hard to use existing technologies. From a distance, 802.1X looks a lot like a RADIUS AAA setup. 802.1X changes the names of some of the components, as shown in Figure 11.30. Compare this to Figure 11.23 to get the new names (the jobs don't change).



• **Figure 11.30** 802.1X components

802.1X combines the RADIUS-style AAA with EAP versions to make a complete authentication solution. The folks who developed 802.1X saw it as a total replacement for every other form of authentication (even Kerberos), but the reality is that most people don't like changing something that already works. To that end, only wireless networking broadly adopted 802.1X.

Technically, wireless networks don't use EAP. They use 802.1X, which in turn uses EAP.

We're not done with authentication and authorization, but at least you now understand the basics of the popular authentication and authorization protocols and standards. There are more protocols to learn, but all of them are rather specialized for specific uses and thus are covered at various places throughout the book.

Encryption Standards

The Internet had authentication long before it had encryption. As a result, almost all encryption came out as a knee-jerk reaction to somebody realizing that his or her TCP/IP application wasn't secure. For years there were new secure versions of just about every protocol in existence. New versions of all the classics started to appear, almost all starting with the word "Secure": Secure FTP, Secure SMTP, and even Secure POP were developed. They worked, but there were still hundreds of not-yet-secured protocols and the specter of redoing all of them was daunting. Fortunately, some new, all-purpose encryption protocols were developed that enabled a client to connect to a server in a secure way while still using their older, unsecure protocols—and it all started because of Telnet.

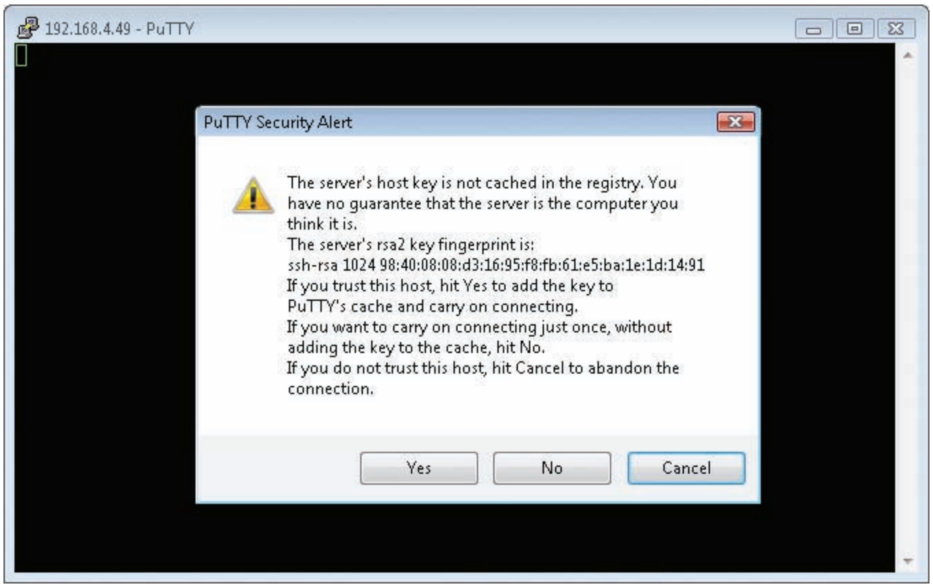
SSH

The broad adoption of the Internet by the early 1990s motivated programmers to start securing their applications. Telnet had a big problem. It was

SSH servers listen on TCP port 22.

incredibly useful and popular, but it was completely insecure. If any one TCP application needed fixing then the world was behind anyone who could fix Telnet. As the story goes, Tatu Ylonen of the Helsinki University of Technology, reacting to an attack that intercepted Telnet user names and passwords on his network, invented a new secure replacement for Telnet called **Secure Shell (SSH)**. You've already seen SSH in action (in Chapter 9, "TCP/IP Applications") as a secure version of Telnet, but now that you know more about security, let's look at SSH in detail.

SSH servers use PKI in the form of an RSA key. The first time a client tries to log into an SSH server, the server sends its public key to the client (Figure 11.31).

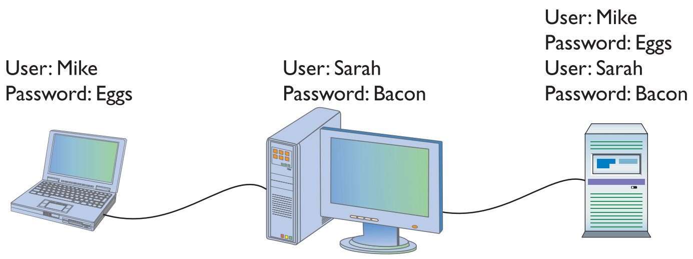


The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 1024 98:40:08:08:d3:16:95:f8:fb:61:e5:ba:1e:1d:14:91
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

• **Figure 11.31** PuTTY getting an RSA key

After the client receives this key, it creates a session ID, encrypts it using the public key, and sends it back to the server. The server decrypts this session ID and uses it in all data transfers going forward. Only the client and the server know this session ID. Next, the client and server negotiate the type of encryption to use for the session. These days AES is popular, but older symmetric-key ciphers such as 3DES may still be used. The negotiation for the cipher is automatic and invisible to the user.

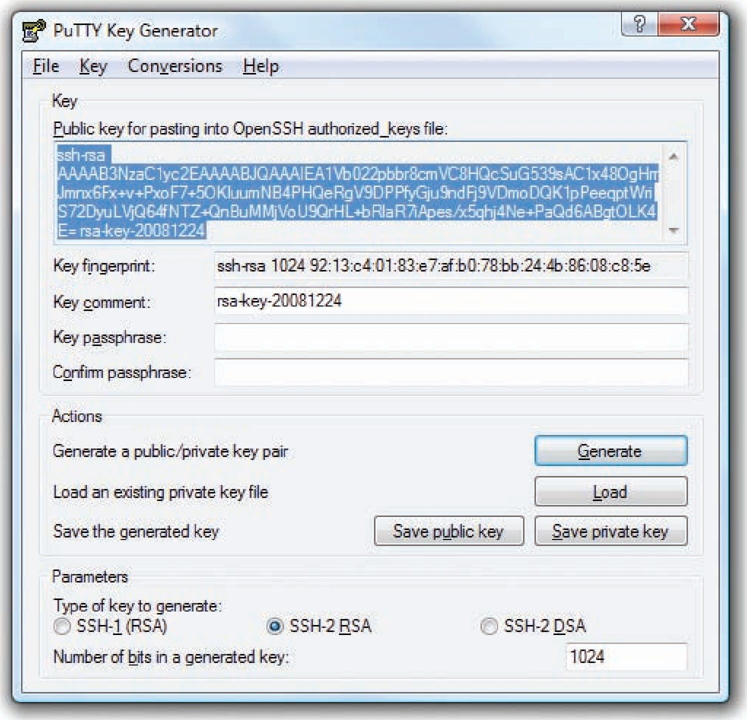
Using RSA and a cipher makes a very safe connection, but the combination doesn't tell the server who is using the client. All SSH servers, therefore, add user names and passwords to authenticate the client (Figure 11.32). Once a user logs in with a user name and password, he or she has access to the system.



• **Figure 11.32** Users on an SSH server

In addition to using a password for authentication, SSH also can use public keys to identify clients. This opens up some interesting possibilities such as no-interaction logins. You can also turn off password login altogether, hardening your server even further. To use public/private keys for authentication, you must first generate a pair of RSA or Digital Signature Algorithm (DSA) keys with a tool such as PuTTYgen (Figure 11.33). The public key is then copied to the server and the private key is kept safe on the client. When you connect to the server, your client generates a signature using its private key and sends it to the server. The server then checks the signature with its copy of the public key, and if everything checks out, you will be authenticated with the server.

If SSH stopped here as a secure replacement for Telnet, that would be fantastic, but SSH has another trick up its sleeve: the capability to act as a *tunnel* for *any* TCP/IP application. Let's see what tunnels are and how they work for us.


PuTTY Key Generator

Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIEA1Vb022pbbr8cmVC8HQcSuG539sAC1x48OgHm
Jmnx6Fx+v+PxoF7+5OKluumNB4PHQeRgV9DPPfyGju9ndFj9VDmoDQK1pPeeqptWn
S72DyuLVjQ64fNTZ+QnBuMMjVoU9QrHL+bRlaR7iApes/x5qhj4Ne+PaQd6ABgtOLK4
E= rsa-key-20081224
```

Key fingerprint: ssh-rsa 1024 92:13:c4:01:83:e7:af:b0:78:bb:24:4b:86:08:c8:5e

Key comment: rsa-key-20081224

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

Parameters

Type of key to generate: SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024

• **Figure 11.33** Generated keys in PuTTYgen

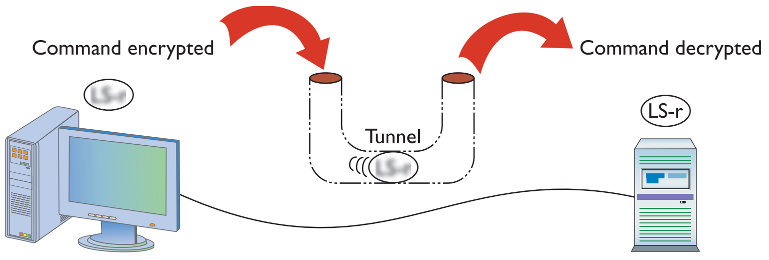
Tunneling

Simply, a **tunnel** is an encrypted link between two programs on two separate computers. Let's take a look at an SSH link between a server and a client. Once established, anything you enter into the client application is encrypted, sent to the server, decrypted, and then acted upon (Figure 11.34).

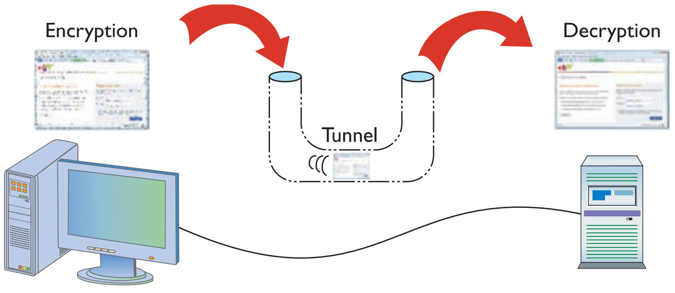
The nature of SSH is such that it took very little to extend the idea of SSH to accept input from any source, even another program (Figure 11.35). As long as the program could redirect to the SSH client and then the SSH server redirect to the server application, anything can go through an SSH connection encrypted. This is an SSH tunnel.

SSH tunnels are wildly popular and fairly easy to set up. Equally, all of the popular SSH clients and servers are designed to go into tunnel mode, usually with no more than the simple click of a check box (Figure 11.36).

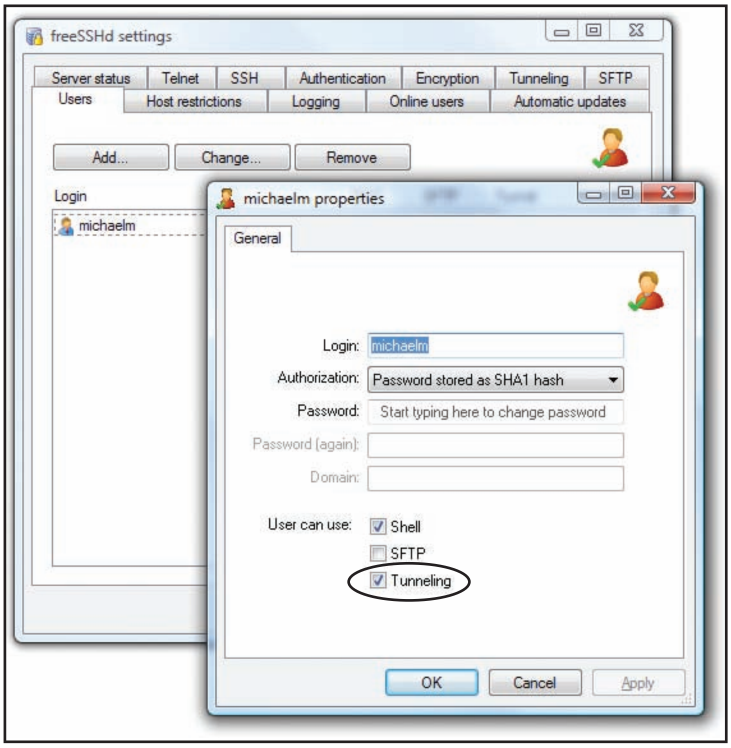
There are many tunneling protocols and standards used in TCP/IP. SSH is one of the simplest types of tunnels so it's a great first exposure to tunneling. As the book progresses, you'll see more tunneling protocols, but you have the basics of tunneling. For now, make sure you understand that a tunnel is an encrypted connection between two endpoints. Any packet that enters the encrypted tunnel, including a packet with unencrypted data, is automatically encrypted, goes through the tunnel, and is decrypted on the other endpoint.



• **Figure 11.34** SSH in action



• **Figure 11.35** Encrypting a Web client



Login: michaelm
Authorization: Password stored as SHA1 hash
Password: Start typing here to change password
Password (again):
Domain:
User can use: [✓ Shell] SFTP [✓ Tunneling]

• **Figure 11.36** Turning on tunneling in freeSSHd server

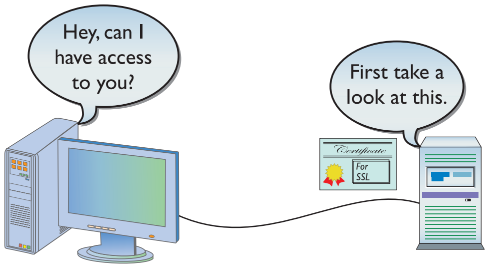
SSH may be popular, but it's not the only option for encryption. All of the other encryption standards are built into combined authentication/encryption standards, as covered in the next section.

Combining Authentication and Encryption

The rest of the popular authentication and encryption standards are combined to include both authentication and encryption in a single standard. Lumping together authentication and encryption into the same standard does not make it weaker than the standards already discussed. These are some of the most popular standards used on the Internet today, because they offer excellent security.

SSL/TLS

The introduction and rapid growth of e-commerce on the World Wide Web in the mid 1990s, made it painfully obvious that some form of authentication and encryption was needed. Netscape Corporation took the first shot at a new standard. At the time, the dominant Web browser was Netscape Navigator. Netscape created a standard called **Secure Sockets Layer (SSL)**. SSL requires a server with a certificate. When a client requests access to an SSL-secured server, the server sends to the client a copy of the certificate. The SSL client checks this certificate (all Web browsers come with an exhaustive list of CA root certificates preloaded), and if the certificate checks out, the server is authenticated and the client negotiates a symmetric-key cipher for use in the session (Figure 11.37). The session is now in a very secure encrypted tunnel between the SSL server and the SSL client.



• **Figure 11.37** SSL at work

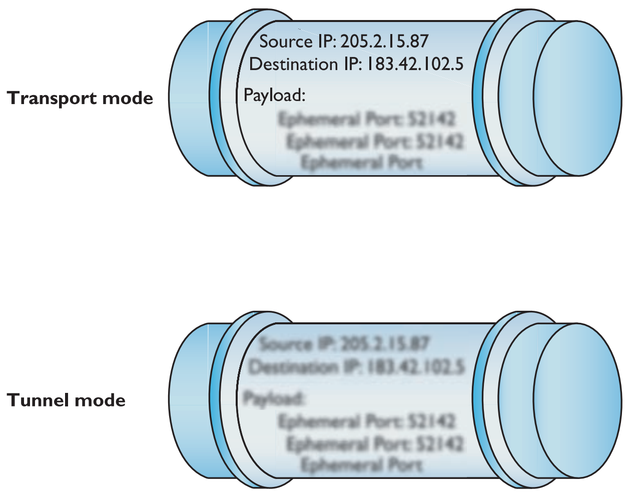
SSL has now been greatly updated to a new standard called **Transport Layer Security (TLS)**. TLS is very similar to SSL, working in almost the same way. TLS is more robust and flexible and works with just about any TCP application. SSL is limited to HTML, FTP, SMTP, and a few older TCP applications. TLS has no such restrictions and is used in securing Voice over IP (VoIP) and virtual private networks (VPNs), but it is still most heavily used in securing Web pages. Every Web browser today uses TLS for HTTPS-secured Web sites, and EAP-TLS is common for more-secure wireless networks.

SSL/TLS also supports mutual authentication, but this is relatively rare.

IPSec

Every authentication and encryption protocol and standard you've learned about so far works *above* the Network layer of the OSI seven-layer model. **Internet Protocol Security (IPSec)** is an encryption protocol that works at the Network layer and, while not very popular currently, will soon become the dominant encryption protocol as IPv6 begins to roll out in the next few years. (See Chapter 13, "IPv6," for details on the upcoming IPv6.)

IPSec works in two different modes: Transport mode and Tunnel mode. In Transport mode, only the actual payload of the IP packet is encrypted: the destination and source IP addresses, port numbers, and other IP header information is still readable. In Tunnel mode, the entire IP packet is encrypted and then placed into an IPSec endpoint where it is encapsulated inside another IP packet. The mode you use depends on the application (Figure 11.38). IPv6 will use the IPSec Transport mode by default.



• **Figure 11.38** IPSec's two modes

IPSec has a strong authentication. Not only does IPSec support Internet Key Exchange (IKE, the standard for exchanging keys on the Internet), it aggressively protects against attacks by adding an authentication header (AH) to every IP packet that use IPSec. While a full discussion of the AH is way beyond the scope of the CompTIA Network+ exam, you should understand that the AH's two different hash values ensure the integrity of every packet.

IPSec is an incredibly powerful authentication/encryption protocol, but until IPv6 is widely implemented it's only common current use is creating secure tunnels between two computers: a job it performs very well. Keep an eye out for IPSec!

Secure TCP/IP Applications

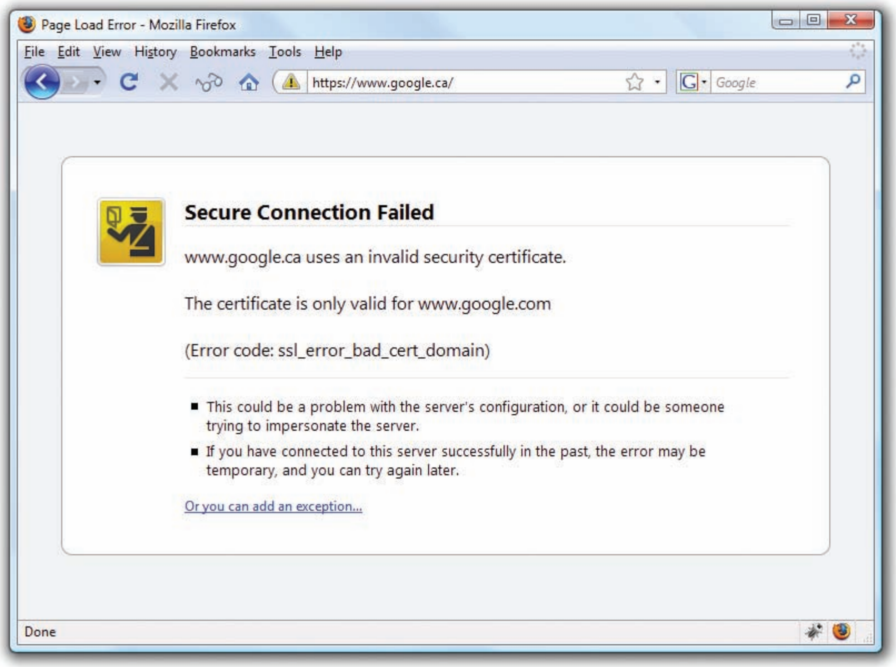
We've covered quite a few TCP/IP security standards and protocols thus far in the chapter, but we really haven't put anything to work yet. It's now time to talk about actual applications that use these tools to make secure connections. As mentioned earlier, this is in no way a complete list, as there are thousands of secure TCP applications; we'll stick to ones you will see on the CompTIA Network+ exam. Even within that group, I've saved discussion of some of the applications for other chapters that deal more directly with certain security aspects (such as remote connections).

Of all the protocols and standard discussed so far, only SSH is also an application.

HTTPS

You've already seen HTTPS back in Chapter 9 so let's do a quick review and then take the coverage a bit deeper. You know that HTTPS documents are unique pages that traditionally start with https:// and that most browsers also show a small lock icon in the lower-right corner or in the address bar. You also know that HTTPS uses SSL/TLS for the actual authentication and encryption process. In most cases, all of this works very well, but what do you do when HTTPS has trouble?

Since you won't get an HTTPS connection without a good certificate exchange, the most common problems are caused by bad certificates. When a certificate comes in from an HTTPS Web site, your computer checks the expiration date to verify the certificate is still valid and checks the Web site's


Page Load Error - Mozilla Firefox
File Edit View History Bookmarks Tools Help
https://www.google.ca/
Google
Done

Secure Connection Failed

www.google.ca uses an invalid security certificate.

The certificate is only valid for www.google.com

(Error code: ssl_error_bad_cert_domain)

- This could be a problem with the server's configuration, or it could be someone trying to impersonate the server.
- If you have connected to this server successfully in the past, the error may be temporary, and you can try again later.

Or you can add an exception...

• **Figure 11.39** Certificate problem

URL to make sure it's the same as the site your are on. If either of these is not correct, you get an error such as shown in Figure 11.39.

If you get one of these errors, you need to decide what to do. Good certificates do go bad (this even happened on my own Web site once) and sometimes the URLs on the certificates are not exactly the same as the site using them. When in doubt, stop. On the other hand, if the risk is low (for example, you're not entering a credit card number or other sensitive information) and you know and trust the site, proceeding is safe in most cases. A courtesy e-mail or phone call to the Web site administrator notifying him or her about the invalid certificate is usually greatly appreciated.

Invalid certificates aren't the only potential problems. After this basic check, the browser checks to see if the certificate has been revoked. Root authorities, like VeriSign, generate Certificate Revocation Lists (CRLs) that a Web browser can check against. Certificates are revoked for a number of reasons, but most of the time the reasons are serious, such as a hacked certificate. If you get a revoked certificate error, it's better to stay away from the site until they fix the problem.

SCP

One of the first SSH-enabled programs to appear after the introduction of SSH was **Secure Copy Protocol (SCP)**. SCP was one of the first protocols used to transfer data securely between two hosts and thus might have replaced FTP. SCP works well but lacks features such as a directory listing. SCP still exists, especially with the well-known UNIX SCP command-line utility, but has for the most part been replaced by the more powerful SFTP.

SFTP

Secure FTP (SFTP), also called SSH FTP, was designed as a replacement for FTP after many of the inadequacies of SCP (such as the inability to see the files on the other computer) were discovered. Although SFTP and FTP have similar

Cross Check

FTP and TFTP

You saw FTP and TFTP back in Chapter 9, "TCP/IP Applications," so check your memory now. How do they differ from SFTP? Do they use the same ports? Would you use FTP and TFTP in the same circumstances? Finally, what's the difference between active and passive FTP?

names and perform the same job of transferring files, they way in which they do that job differs greatly.

The introduction of SSH made it easy to secure most TCP applications just by running them in an SSH tunnel. But FTP was a different case. FTP uses two ports, 20 and 21, creating a two-session communication. This makes FTP a challenge to run in its original form over SSH, because SSH can only handle one session per tunnel. To fix this, a group of programmers from the OpenBSD organization developed a series of secure programs known collectively as **OpenSSH**. SFTP was one of those programs. SFTP looks like FTP, with servers and clients, but relies on an SSH tunnel. If you are on Windows and would like to connect with an SFTP server, WinSCP and FileZilla are two great client options.

SNMP

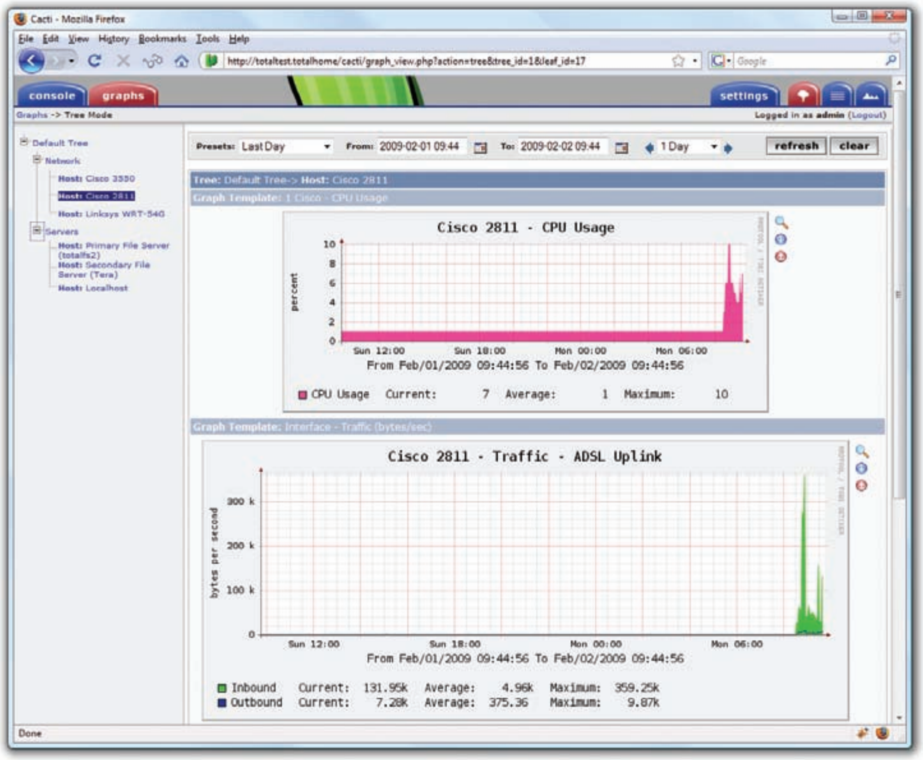
Simple Network Management Protocol (SNMP) is a very popular method for querying the state of SNMP-capable devices. SNMP can tell you a number of settings like CPU usage, network utilization, and detailed firewall hits. SNMP uses *agents* (special client programs) to collect network information from a **Management Information Base (MIB)**, SNMP's version of a server. To use SNMP you need SNMP-capable devices and some tool to query them. One tool is Cacti (www.cacti.net), shown in Figure 11.40. Cacti, like most good SNMP tools, enables you to query an SNMP-capable device for hundreds of different types of information.

SNMP runs on UDP port 161.

SNMP is a useful tool for network administrators, but the first version, SNMPv1, sent all data, including the passwords, unencrypted over the network. SNMPv2 had good encryption, but was rather challenging to use. SNMPv3 is the standard version used today and combines solid, fairly easy-to-use authentication and encryption.

NTP

The **Network Time Protocol (NTP)** does one thing: it gives you the current time. NTP is an old protocol and isn't in and of itself much of a security risk unless you're using some timestamping protocol like Kerberos. Windows is by far the most common Kerberos user, so just make sure all of your computers have access to an NTP server so that users don't run into problems when logging in. NTP uses UDP port 123.



http://totaltest.totalhome/cacti/graph_view.php?action=tree&tree_id=1&leaf_id=17

Presets: Last Day From: 2009-02-01 09:44 To: 2009-02-02 09:44 1 Day

Tree: Default Tree-> Host: Cisco 2811

Graph Template: 1 Cisco - CPU Usage

```
Cisco 2811 - CPU Usage
From Feb/01/2009 09:44:56 To Feb/02/2009 09:44:56
CPU Usage   Current:      7   Average:      1   Maximum:     10
```

Graph Template: Interface - Traffic (bytes/sec)

```
Cisco 2811 - Traffic - ADSL Uplink
From Feb/01/2009 09:44:56 To Feb/02/2009 09:44:56
Inbound    Current:  131.95k   Average:    4.96k   Maximum:  359.25k
Outbound   Current:    7.28k   Average:   375.36   Maximum:    9.87k
```

• **Figure 11.40** Cacti at work

Chapter 11 Review

Chapter Summary

After reading this chapter and completing the exercises, you should understand the following about networking.

Discuss the standard methods for securing TCP/IP networks

- TCP/IP security can be broken down into four areas: encryption, nonrepudiation, authentication, and authorization.
- Encryption means to scramble, to mix up, to change the data in such a way that bad guys can't read the data.
- Nonrepudiation is the process that guarantees that the data is as originally sent and that it came from the source you think it should have come from.
- Authentication means to verify that whoever accesses the data is the person you want accessing that data.
- Authorization defines what a person accessing the data can do with that data.
- All data starts as plaintext (also called cleartext), meaning the data is in an easily read or viewed industry-wide standard format.
- A cipher is a series of complex and hard-to-reverse mathematics—called an algorithm—you run on a string of ones and zeroes to make a new set of seemingly meaningless ones and zeroes. More specifically, a cipher is a general way to encrypt data, an algorithm is the cipher's underlying mathematical formula, and a complete algorithm is the implementation of the cipher.
- A symmetric-key algorithm is any encryption algorithm that uses the same key for both encryption and decryption. There are two types of symmetric-key algorithms: block ciphers and stream ciphers.
- Block ciphers encrypt data in single chunks of a certain length. Stream ciphers encrypt a single bit at a time.
- Data Encryption Standard (DES) is the oldest TCP/IP symmetric-key algorithm and uses a 64-bit block with a 56-bit key. DES is susceptible to brute-force attacks.
- Advanced Encryption Standard (AES) is the most secure TCP/IP symmetric-key algorithm and uses a 128-bit block with a 128-, 192-, or 256-bit key. AES is practically uncrackable.
- Symmetric-key encryption has one serious weakness: anyone who gets a hold of the key can encrypt or decrypt.
- Public-key cryptography is an implementation of asymmetric-key encryption, which uses one key to encrypt and a different key to decrypt.
- A key pair consists of a public key, which is shared and distributed to senders to use to encrypt data, and a private key, which is kept only by the recipient and used to decrypt data.
- A hash is a mathematical function that you run on a string of binary digits of any length that results in a value of some fixed length, often called a checksum or a digest.
- A cryptographic hash function is a one-way function that produces a unique checksum that can be used to verify nonrepudiation. MD5 and SHA are popular hashes for this type of work.
- A digital signature is a string of ones and zeroes that can only be generated by the sender and is another form of nonrepudiation.
- A certificate is a standardized type of digital signature used to verify the identity of someone (or something) you do not know, like a Web site. A certificate usually includes the digital signature of a third party, a person or a company that guarantees that who is passing out this certificate truly is who they say they are. VeriSign and thawte are popular certificate authorities.
- An access control list (ACL) is used to control authorization, or what a user is allowed to do once they have been authenticated. There are three types of ACL access modes: MAC, DAC, and RBAC.
- In a mandatory access control (MAC) security model, every resource is assigned a label that defines its security level. If the user lacks that security level, they do not get access.
- Discretionary access control (DAC) is based on the idea that there is an owner of a resource who may

at his or her discretion assign access to that resource.

- Role-based access control (RBAC) is the most popular model used in file sharing and defines a user's access to a resource based on the user's group membership.

Compare TCP/IP security standards

- The Point-to-Point Protocol (PPP) enables two point-to-point devices to connect, authenticate with a user name and password, and negotiate the network protocol the two devices will use.
- PPP includes two methods to authenticate a user name and password: PAP and CHAP.
- Password Authentication Protocol (PAP) transmits the user name and password over the connection in plaintext, which is not secure.
- Challenge Handshake Authentication Protocol (CHAP) provides a more secure authentication routine because it relies on hashes based on a shared secret, usually a password that both ends of the connection know. Microsoft created its own version called MS-CHAP.
- Authentication, Authorization, and Accounting (AAA) is a philosophy applied to computer security. RADIUS and TACACS+ are standard implementations of AAA.
- Remote Authentication Dial-In User Service (RADIUS) is the better known of the two AAA standards and was created to support ISPs with hundreds if not thousands of modems in hundreds of computers to connect to a single central database.
- Microsoft's RADIUS server is called Internet Authentication Service (IAS) and comes built in with most versions of Microsoft Windows Server. FreeRADIUS is a popular RADIUS server for UNIX/Linux.
- Terminal Access Controller Access Control System Plus (TACACS+) is a proprietary protocol developed by Cisco to support AAA in a network with many routers and switches.
- Kerberos is an authentication protocol for TCP/IP networks with many clients all connected to a single authenticating server, unlike PPP.
- Kerberos, which is the authentication protocol for all Windows networks using a domain controller, uses a Key Distribution Center (KDC) that has two processes: the Authentication Server (AS) and the Ticket-Granting Service (TGS).
- The Authentication Server authenticates users at login and, if successful, sends a Ticket-Granting Ticket (TGT) (good for eight hours by default) allowing the user to access network resources without having to reauthenticate.
- The timestamped TGT is sent to the TGS, which returns an access token used by the client for authorization to network resources.
- The Extensible Authentication Protocol (EAP) was developed to help two devices negotiate the authentication process. It is used primarily in wireless networks. There are six commonly used types of EAP: EAP-PSK, EAP-TLS, EAP-TTLS, EAP-MS-CHAPv2 (PEAP), EAP-MD5, and LEAP.
- EAP Personal Shared Key (EAP-PSK) is the most popular form of authentication used in wireless networks today.
- Early wireless networks lacked any form of authentication, so the wireless community grabbed a preexisting authentication standard called 802.1X to use in their wireless networks. 802.1X combines the RADIUS-style AAA with EAP versions to make a complete authentication solution.
- Secure Shell (SSH) is a secure replacement for Telnet. SSH uses PKI in the form of an RSA key. At login, the SSH server sends its public key to the client. The client then encrypts data using the public key and transmits the data, which is subsequently decrypted on the server with the private key.
- Netscape created the Secure Sockets Layer (SSL) standard, which requires a server with a certificate. SSL has been updated to the Transport Layer Security (TLS) standard and is used for secure Web transactions, such as online credit card purchases.
- SSL is limited to HTML, FTP, SMTP, and a few older TCP applications while TLS is less restrictive and is used for everything SSL does in addition to VoIP and VPNs.
- IPSec is an encryption protocol and is destined to become the dominant encryption protocol under IPv6. IPSec works in two different modes: Transport mode and Tunnel mode. IPv6 uses the IPSec Transport mode by default.
- In Transport mode, only the actual payload of the IP packet is encrypted; the destination and source

IP addresses, port numbers, and other IP header information is still readable.

- In Tunnel mode, the entire IP packet is encrypted and then placed into an IPSec endpoint where it is encapsulated inside another IP packet.

Implement secure TCP/IP applications

- HTTPS uses SSL/TLS for the actual authentication and encryption process. Most browsers show a small lock icon in the lower right corner or in the address bar when an HTTPS connection is established.
- The most common problems with HTTPS connections are caused by bad or outdated certificates.
- Secure Copy Protocol (SCP) is an SSH-enabled program or protocol used to securely copy files between a client and a server. It has been replaced by Secure FTP (SFTP).
- Simple Network Management Protocol (SNMP) is a method for querying the state of SNMP-capable devices. SNMP can tell you a number of settings like CPU usage, network utilization, and detailed firewall hits. SNMP uses agents and MIBs to capture and monitor network usage.
- SNMPv1 sent all data, including the passwords, unencrypted over the network. SNMPv2 had good encryption but was rather challenging to use. SNMPv3 is the standard version used today and combines solid, fairly easy-to-use authentication and encryption.
- Network Time Protocol (NTP) gives you the current time. It isn't much of a security risk unless you're using some timestamping protocol like Kerberos.

Key Terms

802.1X *(299)*
access control list (ACL) *(291)*
Advanced Encryption Standard (AES) *(283)*
algorithm *(280)*
asymmetric-key algorithm *(282)*
authentication *(279)*
Authentication Server (AS) *(297)*
Authentication, Authorization, and Accounting (AAA) *(279)*
authorization *(279)*
block cipher *(282)*
certificate *(287)*
Challenge Handshake Authentication Protocol (CHAP) *(293)*
cipher *(280)*
ciphertext *(281)*
cleartext *(280)*
complete algorithm *(280)*
Data Encryption Standard (DES) *(282)*
digital signature *(287)*
discretionary access control (DAC) *(291)*
encryption *(279)*
Extensible Authentication Protocol (EAP) *(297)*
FreeRADIUS *(296)*
hash *(286)*
Internet Authentication Service (IAS) *(296)*
Internet Protocol Security (IPSec) *(302)*
Kerberos *(296)*
Key Distribution Center (KDC) *(297)*
key pair *(284)*
Lightweight Extensible Authentication Protocol (LEAP) *(298)*
Management Information Base (MIB) *(305)*
mandatory access control (MAC) *(291)*
MD5 *(286)*
MS-CHAP *(293)*
Network Access Server (NAS) *(295)*
Network Time Protocol (NTP) *(305)*
nonrepudiation *(279)*
OpenSSH *(305)*
Password Authentication Protocol (PAP) *(293)*
plaintext *(280)*
Point-to-Point Protocol (PPP) *(292)*
Protected Extensible Authentication Protocol (PEAP) *(298)*
public-key cryptography *(283)*
public-key infrastructure (PKI) *(290)*
Remote Authentication Dial-In User Service (RADIUS) *(295)*
Rivest Cipher 4 (RC4) *(283)*

Rivest Shamir Adleman (RSA) *(283)*
role-based access control (RBAC) *(291)*
Secure Copy Protocol (SCP) *(304)*
Secure FTP (SFTP) *(304)*
Secure Hash Algorithm (SHA) *(287)*
Secure Shell (SSH) *(300)*
Secure Sockets Layer (SSL) *(302)*
Simple Network Management Protocol (SNMP) *(305)*
stream cipher *(282)*
symmetric-key algorithm *(282)*
Terminal Access Controller Access Control System Plus (TACACS+) *(296)*
Ticket-Granting Ticket (TGT) *(297)*
Transport Layer Security (TLS) *(302)*
tunnel *(301)*

Key Term Quiz

Use the Key Terms list to complete the sentences that follow. Not all the terms will be used.

1. _______________ defines what a person accessing data can do with that data.
2. _______________ is the act of verifying you are who you say you are.
3. _______________ is the process of guaranteeing that data is as originally sent and that it came from the source from which you think it should have come.
4. A(n) _______________ encrypts data in fixed-length chunks at a time.
5. _______________ is a secure replacement for Telnet.
6. A(n) _______________ uses one key to encrypt data and a different key to decrypt the same data.
7. SSL has been replaced by the more robust _______________.
8. SCP has been replaced by _______________, a secure protocol for copying files to a server.
9. _______________ is the default authentication protocol for Windows domains and is extremely time sensitive.
10. _______________ uses a 128-bit block, up to a 256-bit key, and is a virtually uncrackable encryption algorithm.

Multiple-Choice Quiz

1. Justin wants his team to be able to send him encrypted e-mails. What should he do?
   - **A.** Send to each team member his private key.
   - **B.** Send to each team member his public key.
   - **C.** Ask each team member for their private key.
   - **D.** Ask each team member for their public key.
2. Which of the following are popular cryptographic hashing functions? (Select two.)
   - **A.** MD5
   - **B.** SHA
   - **C.** RADIUS
   - **D.** TACACS+
3. A public and private key pair is an example of what?
   - **A.** Symmetric-key algorithm
   - **B.** Asymmetric-key algorithm
   - **C.** Certificate
   - **D.** RADIUS
4. Which authentication protocol is time sensitive and is the default authentication protocol on Windows domains?
   - **A.** PPP
   - **B.** MS-CHAP
   - **C.** IPSec
   - **D.** Kerberos
5. What helps to protect credit card numbers during online purchases? (Select two.)
   - **A.** Certificates
   - **B.** TLS
   - **C.** SCP
   - **D.** NTP

6. Emily wants to remotely and securely enter commands to be run at a remote server. What application should she use?
   - **A.** Telnet
   - **B.** SSH
   - **C.** SFTP
   - **D.** RSA
7. A hash function is by definition:
   - **A.** A complex function
   - **B.** A PKI function
   - **C.** A one-way function
   - **D.** A systematic function
8. In order to have a PKI infrastructure you must have a(n):
   - **A.** Web server
   - **B.** Web of trust
   - **C.** Root authority
   - **D.** Unsigned certificate
9. Which term describes the process of guaranteeing that data that is received is in fact the data that was sent—and that it came from the presumed source?
   - **A.** Authentication
   - **B.** Authorization
   - **C.** Encryption
   - **D.** Nonrepudiation
10. If you saw some traffic running on TCP port 49, what AAA standard would you know was running?
   - **A.** PPP
   - **B.** RADIUS
   - **C.** MS-CHAP
   - **D.** TACACS+
11. What is the difference between RADIUS and TACACS+?
   - **A.** RADIUS is authentication control for Windows networks while TACACS+ is authentication control for UNIX/Linux networks.
   - **B.** RADIUS is an implementation of authentication control while TACACS+ is an implementation of authorization control.
   - **C.** RADIUS is a generic name for authentication control, and there are implementations for Windows, UNIX, and Linux servers. TACACS+ is authentication control for Cisco routers and switches.
   - **D.** RADIUS supports encryption, TACACS+ does not and is therefore less desirable in a network.
12. AES is a(n) __________ cipher.
   - **A.** Block
   - **B.** Forwarding
   - **C.** Stream
   - **D.** Asymmetric
13. Which authentication protocol is broadly used on wireless networks?
   - **A.** 802.1X
   - **B.** PPP
   - **C.** PAP
   - **D.** MS-CHAP
14. Digital signatures and certificates help which aspect of computer security?
   - **A.** Accounting
   - **B.** Authentication
   - **C.** Authorization
   - **D.** Nonrepudiation
15. Which authorization model grants privileges based on the group membership of network users?
   - **A.** MAC
   - **B.** DAC
   - **C.** RBAC
   - **D.** GAC

Essay Quiz

1. Explain the difference between symmetric-key and asymmetric-key algorithms and give examples of each. Which is more secure? Why?
2. Access control lists help to control the authorization of network resources. Explain the differences between the three ACL access models.
3. You receive a call from a distressed user telling you they were in the middle of an online purchase (just entering their credit card number) when they noticed a certificate warning on the screen saying the Web site's certificate has expired. What advice would you give the user?

Lab Projects

• Lab Project 11.1

Download a copy of GnuPG from www.gnupg.org and one of the frontends from www.gnupg.org/related_software/frontends.en.html. Generate a key pair and share your public key with a classmate. Have your classmate encrypt a file using your public key and e-mail it to you. Decrypt your file with your private key.

• Lab Project 11.2

You have learned many acronyms in this chapter! Make a list of the following acronyms, state what they stand for, and briefly describe them. Use this as a study sheet for the CompTIA Network+ certification exam: DES, AES, RSA, MD5, SHA, PKI, CRAM-MD5, ACL, MAC, DAC, RBAC, PPP, PAP, CHAP, MS-CHAP, AAA, RADIUS, TACACS+, KDC, AS, TGT, SID, EAP, EAP-TLS, EAP-PSK, EAP-TTLS, EAP-MS-CHAPv2, PEAP, EAP-MD5, LEAP, SSH, SSL, TLS, HTTPS, SCP, SFTP, SNMP, and NTP.