Chapter 20

Secure Socket Layer and Transport Layer Security

If you've used your credit card over the Internet, you've probably used Secure Socket Layer (SSL). Its main purpose is to make e-commerce users feel secure about sending their financial information over the Internet.[1] SSL seems to complete the transaction quickly, and much of what goes on in that short time is transparent to the user.

SSL is a data communication *protocol* (set of agreed on procedures) that implements three cryptographic assurances—authentication, confidentiality, and message integrity—and provides secure key exchange between an Internet browser and an Internet server. Note that SSL does not offer nonrepudiation.

Padlock symbolizes SSL session.

To notify users that they're using SSL, current versions of the Netscape and Microsoft Internet browsers display a small padlock (see Figure 20-1). Older versions of Netscape display a blue border around the browser window.

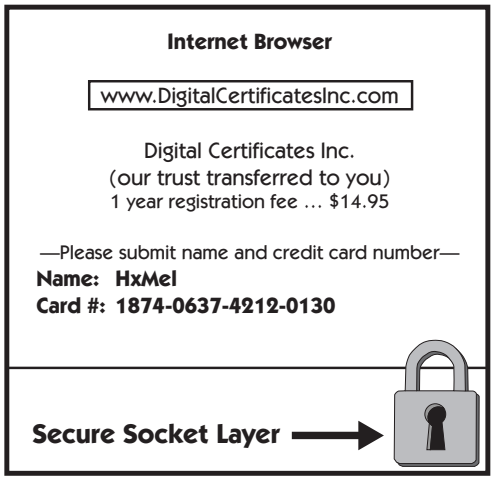

Figure 20-1 The SSL symbol is shown in a typical browser window.

1. SSL is also infrequently used for e-mail, ftp (file transfer protocol), VPN, and so on.

History of SSL

Netscape developed SSL in 1994 and released version 2 in early 1995. By that time, William Gates, Jr., understood the importance of the Internet and quickly developed and released a competing Microsoft product called Private Communicating Technology (PCT); it briefly competed with SSL. Media attention to SSL bugs and PCT forced Netscape to release SSL version 3 (SSL v3) in 1995. PCT is seldom used anymore.

IETF set standard

In 1996 the Internet Engineering Task Force (IETF) formed a committee, the Transport Layer Security (TLS) working group, to develop and publish an SSL standard. In January 1999, the TLS working group published the TLS protocol, which was based on SSL v3. Both Microsoft and Netscape support TLS. Interestingly, Microsoft implemented TLS in its browser before Netscape did.

SSL comes first in the title of this chapter because, as we're writing this book, more people have heard of it than TLS. The differences between SSL v3 and TLS version 1 are minor. Almost everything in this chapter is applicable to SSL v3 as well as TLS. At the end of the chapter we note some differences.

Overview of an SSL Session

Figure 20-2 shows the symbols used in this chapter.

Figure 20-3 presents an overview of an SSL/TLS session. (More SSL/TLS details follow in the next section.)

SSL/TLS allows two computers to negotiate cryptographic parameters.

Negotiating cryptographic parameters between two computers that probably don't know each other's cryptographic capabilities is the first thing SSL/TLS participants must do. (SSL/TLS accommodates a variety of cryptographic choices from which the client and server may choose.) Then Bob uses Alice's public key to send her a secret. They use the secret to independently make the identical secret key. Next, in our example, Bob authenticates Alice, a server merchant, but Alice does not authenticate him. Note that because SSL/TLS uses the agreed-on secret key for authentication, Alice and Bob agree on a secret key before authentication. Finally, they use the secret key to exchange messages.

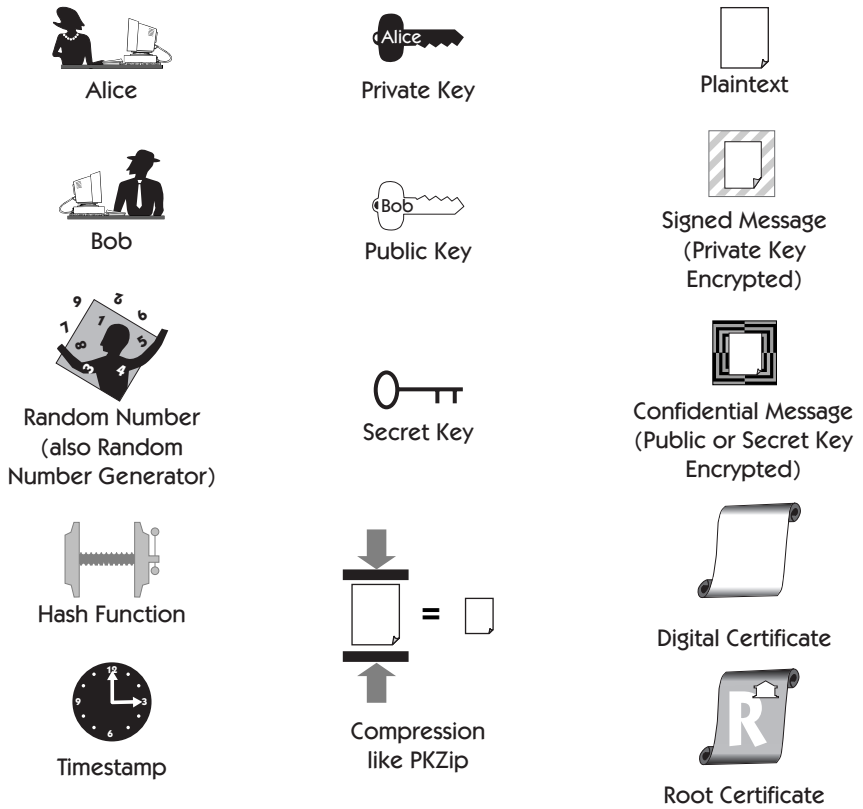

Figure 20-2 These symbols are used to represent the concepts discussed in this chapter.

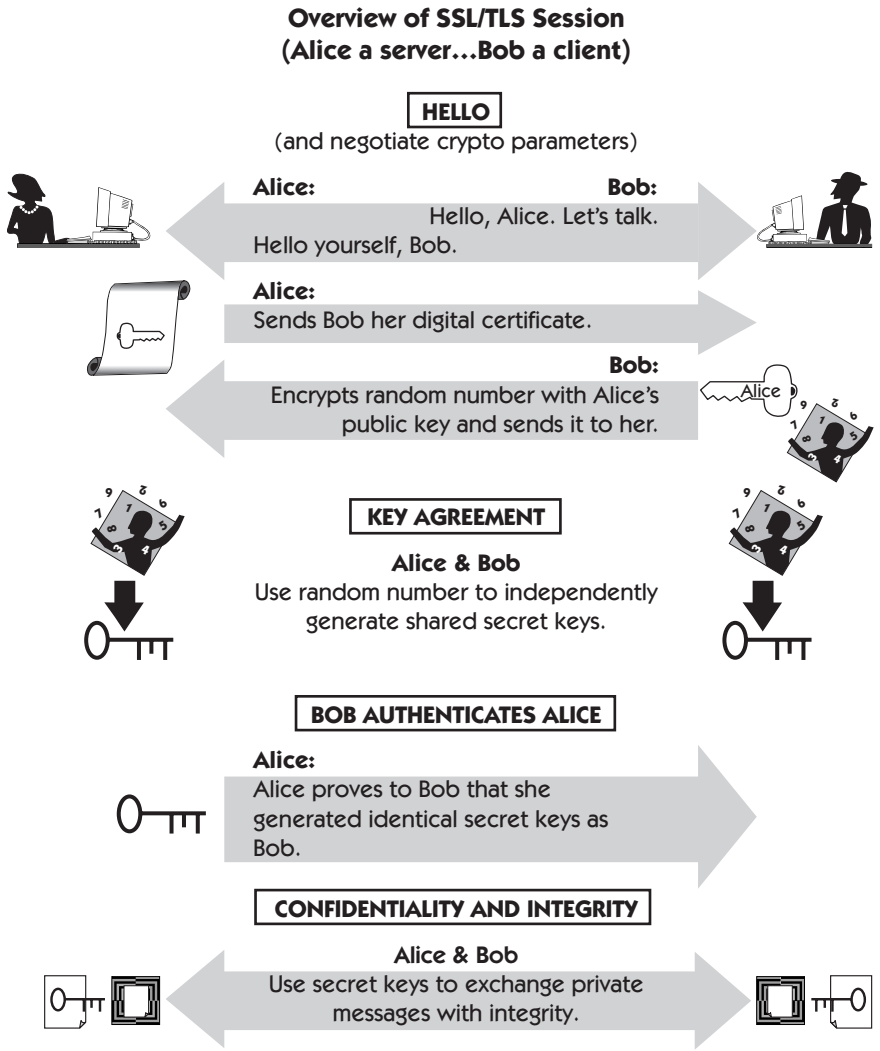

Figure 20-3 Overview of an SSL/TLS session. Bob, a customer, authenticates Alice, a merchant.

An SSL Session in Detail

Let's watch as Bob, an Internet customer using his browser, connects to an Internet server and merchant, AliceDotComStocks. (We'll abbreviate AliceDotComStocks as Alice unless we feel it's worth reinforcing that Alice is the merchant server.) Although SSL/TLS can accommodate many different

cryptographic variations, for simplicity we show only one cryptographic variation in the following example.

Hello and Negotiate Parameters

Bob, the client, initiates contact and suggests some cryptographic parameters.

Alice responds. Definition: Cipher suite

In Figure 20-4, Bob sends a plaintext message `Hello` and suggests some parameters for their conversation. Bob also sends Alice an alternative in case she can't accommodate his first suggestion. For example, if Alice can't do TLS, Bob offers her the option SSL v3. Although we're showing only one alternative, Bob may include many possible alternatives.

Alice responds with her choice of cryptographic parameters. In Figure 20-5, Alice agrees with all of Bob's suggestions except that she requests DES instead of Triple DES. Alice and Bob agree to converse using RSA for key exchange,

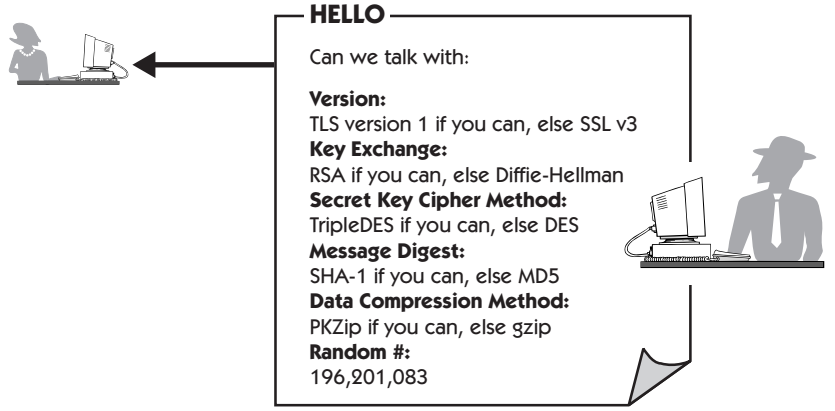

Figure 20-4 Bob sends `Hello` and suggests parameters.

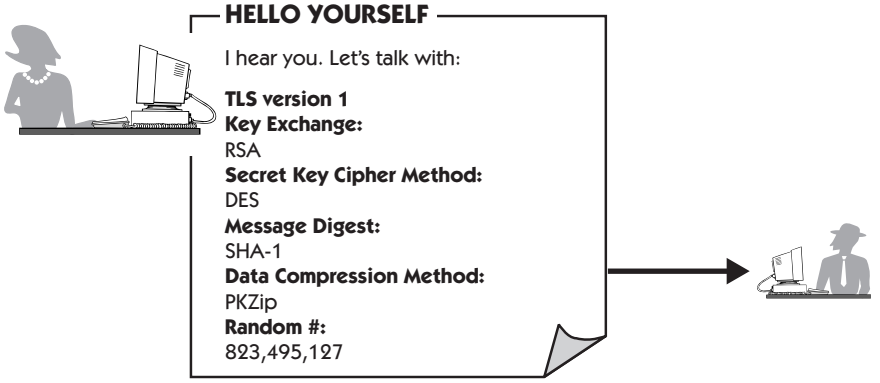

Figure 20-5 Alice responds.

DES for bulk data encryption, and SHA-1 for message digesting. They also agree on a data compression method.[2] Put together, all these agreements are called a *cipher suite*. (It's not likely that Alice couldn't accommodate Triple DES; we're showing it simply as an example of an alternative option. Rijndael isn't offered because it's still too new.)

Exchanging Digital Certificates

SSL allows the client to authenticate the server. Bob authenticates Alice.

After responding to Bob's `Hello`, Alice sends Bob her digital certificate.[3]

In our example, a trusted root CA signed Alice's certificate. Bob uses his trusted copy of that particular root CA's public key to verify Alice's certificate and enclosed public key (see Figure 20-6). Recall from Chapter 17 that Internet browsers install trusted CA certificates (public keys).

Only if Alice sends her digital certificate to Bob does SSL/TLS permit her to request Bob's certificate. She's not required to request Bob's certificate, and even if she does request it, he need not comply. Later in the chapter we show why.

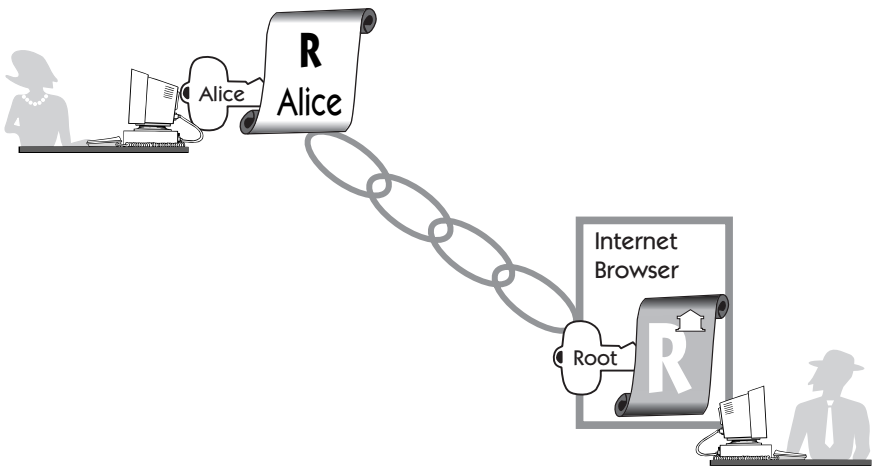

Figure 20-6 Alice sends her digital certificate. Bob uses the trusted CA certificate to verify Alice's digital certificate (her public key).

2. A compression program that also decompresses data, such as PKZip, WinZip, gzip, or StuffIt.
3. And any other digital certificates Bob needs to build a digital certificate chain to Alice's digital certificate.

Key Agreement (Exchange)

Definition: pre-master secret

Bob generates a 48-byte random value called a *pre-master secret* (see Figure 20-7). He encrypts it with Alice's public RSA key and sends it to her. Then Alice decrypts the pre-master secret with her matching private RSA key.

So far, our examples of secret key exchange have indicated that two people need share only one secret key. Real-world systems prefer to use more secret keys to thwart cryptanalysis. In SSL/TLS each end of the connection (Alice and Bob) generates six secret keys.

SSL/TLS uses six separate secret keys.

Alice generates three secret keys for Alice-to-Bob messages, shown on the left in Figure 20-8. The first key (such as a DES secret key) is for encryption, the second key is for the message integrity (HMAC), and the third key is used to initialize the cipher (IV).[4] Note that these keys are used only for messages Alice sends to Bob and not for messages Bob sends to Alice. Alice generates and uses three additional keys for Bob-to-Alice messages. Similarly, because Bob must generate exactly the same keys as Alice, he generates three secret keys for Alice-to-Bob messages and three keys for Bob-to-Alice messages, shown on the right in Figure 20-8.

It's not important for our discussion, but if you're interested, later in the chapter you'll find additional specifics on how the six shared SSL/TLS secret values are generated.

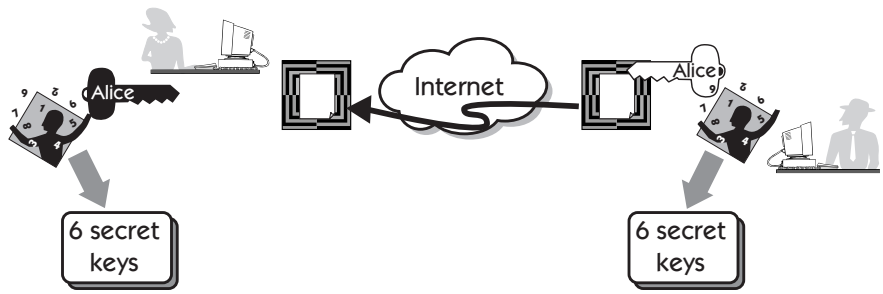

Figure 20-7 Bob generates a 48-byte random number called a pre-master secret. He encrypts it with Alice's public key and sends it to her.

4. Think of cipher initialization as random plaintext that primes the cipher pump. In cryptographic literature it's often abbreviated IV.

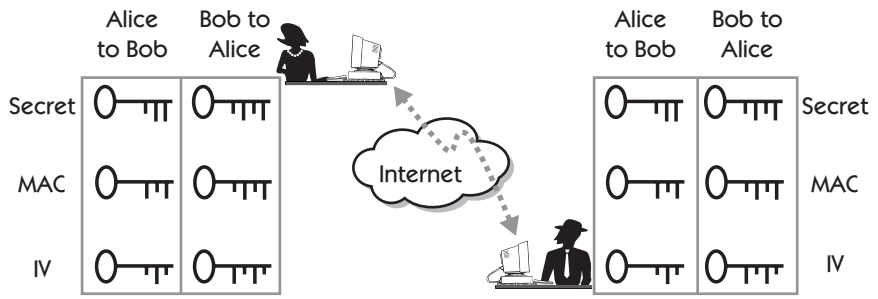

Figure 20-8 Alice's and Bob's identical shared secret keys.

Authentication

AliceDotComStocks (the server merchant) doesn't always authenticate Bob (the client).

It is at this point that you might expect Alice to authenticate Bob, but in our example she doesn't. How can Bob get away without being authenticated? There are two major reasons.

First, Alice will check that Bob's credit card (you know she'll ask for it soon) hasn't been invalidated by the credit card company, and that's all the authentication she currently needs. Also, it takes time for her to authenticate Bob, and she doesn't want an impatient Internet customer to click off her site and cancel the sale.

Second, most e-commerce clients don't yet have digital certificates, and e-commerce merchants are currently assuming the risk entailed. However, Alice and other e-commerce merchants probably won't assume this risk much longer as they get stung by Internet con artists. If you're interested, at the end of the chapter you'll find more specifics on how Alice can authenticate Bob.

Bob authenticates that Alice has independently calculated the identical secret keys.

Bob sends Alice a message encrypted with their shared secret keys. This message, called a *finished handshake* message, is the first message encrypted with the secret keys Bob and Alice independently generated during the key agreement phase.

Alice responds to Bob with her own encrypted finished handshake message. Bob is now assured that he must be communicating with Alice because Bob sent the pre-master secret encrypted with Alice's public RSA key (Figure 20-7). Only Alice could have decrypted the pre-master secret used to calculate the six shared secret keys.

Alice and Bob can now begin to use their six shared secrets for bulk data encryption, such as ordering merchandise with a credit card or insurance forms.

Confidentiality and Integrity

Figure 20-9 shows the general steps Alice takes to prepare an SSL/TLS secure message using the agreed-on cipher suite and secret keys.[5]

1. She compresses the message using the agreed compression method.
2. She hashes the compressed data and her secret HMAC key to make an HMAC.
3. She encrypts the combination of compressed data and HMAC with her Alice-to-Bob DES key.

Recall that the HMAC secret key is different from the DES encryption secret key.

Bob receives the SSL/TLS encrypted message and reverses the process, as shown in Figure 20-10.

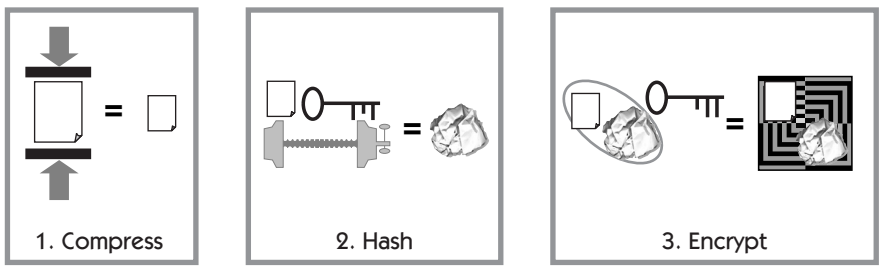

Figure 20-9 Alice prepares an encrypted message for transport.

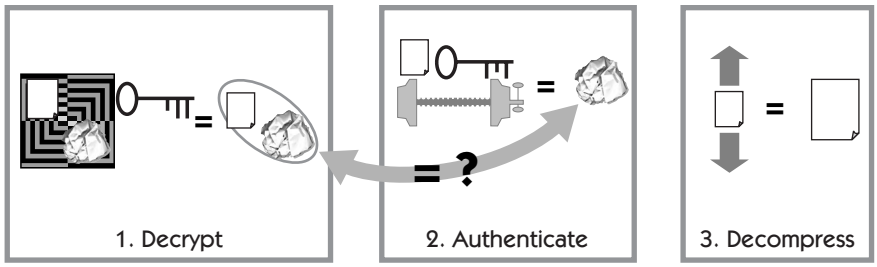

Figure 20-10 Bob verifies and recovers a message from Alice.

5. In this overview, we omit some details such as data fragmentation and padding.

1. He decrypts the combination of compressed data and HMAC with his copy of Alice-to-Bob DES key.
2. He authenticates the message in two steps. First, he hashes the decrypted compressed data and secret HMAC key. Then he compares (see the symbol "= ?") the HMAC from step 1 to the HMAC from this newest hash.
2. He decompresses[6] and recovers the plaintext message.

TLS Variations

The preceding example shows Bob encrypting a random number, the pre-master secret, using Alice's public RSA key. After Alice decrypts it with her private RSA key only Alice and Bob know the pre-master secret used to generate the six shared secret keys.

SSL/TLS can also use Diffie-Hellman (see Chapter 10 and Appendix A) to independently create the shared pre-master secret key and six shared secret keys shown in Figure 20-8. One of the Diffie-Hellman variations doesn't support authentication.

Anonymous Diffie-Hellman

Anonymous Diffie-Hellman doesn't authenticate Bob or Alice.

One variation, called Anonymous Diffie-Hellman, allows Alice and Bob to simultaneously generate six shared secrets without either one authenticating the other. But then neither Bob nor Alice is certain who is on the other side of the connection. Why would anyone use this approach?

As we said, AliceDotComStocks is often satisfied with receiving a credit card number and is willing to assume any risks entailed in someone else's use of Bob's credit card without Bob's authorization.

What about Bob? Bob can decide to trust that he has downloaded the authentic AliceDotComStocks Web page and associated forms. Perhaps it's not a big risk. People often give their credit card numbers to strangers on the telephone, in restaurants, and so on. Should you reveal your credit card number this way? Well, most of us do, and not enough of us have had problems with this to stop using credit cards in this way. But if you're sending something very valuable over the Internet—such as the password to your bank account or personal financial or health records—it's not a good idea to trust Anonymous Diffie-Hellman key agreement.

6. Remember, compression means using a compression-decompression method such as PKZip, WinZip, or StuffIt. Don't confuse this with HMAC, which does not decompress.

Fixed and Ephemeral Diffie-Hellman

Fixed and Ephemeral Diffie-Hellman support authentication.

Both of these Diffie-Hellman variations support authentication. In Fixed (or static) Diffie-Hellman, Alice and Bob exchange their public Diffie-Hellman values using trusted digital certificates. In Ephemeral Diffie-Hellman, Alice and Bob exchange their public Diffie-Hellman values signed with RSA or DSA keys.

Comparing TLS, SSL v3, and SSL v2

SSL v3 and TLS are almost identical.

The TLS standard says that "TLS version 1.0 and SSL v3 are very similar, [and] thus, supporting both is easy." Some of the differences include minor changes in HMAC calculations, cipher suite support, and pseudo-random number calculations. You can think of TLS as "SSL v3.1."

SSL v3 and SSL v2 are very different.

The differences between SSL v3 and SSL v2, however, are more pronounced. No one who has access to SSL v3 should use SSL v2. Arguably, users should turn off support for SSL v2 in their Internet browsers.

A Big Problem with SSL v2

Because of the way SSL v2 negotiates the cipher suite, BlackHat can convince Alice and Bob to use much weaker encryption than they are capable of using. This is called a *cipher suite rollback* attack. For example, even though Alice and Bob can do Triple DES (i.e., 168-bit encryption), BlackHat can attack their SSL v2 cipher suite negotiations and convince them to use something much weaker, such as 40-bit encryption.

A Possible Problem with TLS and SSL

The danger of traffic analysis

No version of TLS or SSL protects against *traffic analysis*. A cryptanalyst doing traffic analysis watches the number of messages sent from and to a particular Internet address. Although the analyst may not know what's inside each message, it's possible that the knowledge of heavier-than-usual message traffic between two addresses can be, by itself, useful information. For example, a heavy exchange of messages between legal offices in separate cities might signal intentions that an adversary could use to his or her advantage.

The threat posed by BlackHat's use of traffic analysis probably doesn't affect most Internet users. Nevertheless, it's a serious enough concern that Chapter 21 shows a way to limit the damage from traffic analysis.[7] SSL and TLS can't.

7. Supplemental tech note: TLS (and SSL) are above the TCP/IP layer in the protocol stack. There's no way they can hide the address and port of the source and destination addresses.

Generating Shared Secrets

Definition: master secret

As promised earlier in this chapter, this section describes some of the steps that occur after Bob sends Alice the pre-master secret. Alice and Bob use the pre-master secret, the random values they exchanged in the `Hello` messages, and a pseudo-random function (PRF) to independently and simultaneously generate a *master secret*.

The PRF is very similar to a message digest function run many times; it adds enough uncertainty to the generation of secrets that it makes replication by an attacker infeasible. After completing two rounds of PRF, Alice and Bob have independently generated six equivalent secrets, as shown in Figure 20-11.

Use both SHA-1 and MD5 for added security.

The PRF uses two different message digest functions because, as stated in the IETF TLS standard document, "In order to make the PRF as secure as possible, it uses two hash algorithms in a way which should guarantee its security if either algorithm remains secure." This means that two different message digest methods are used in case BlackHat cracks one of them; the other hash function, which presumably is secure, ensures that BlackHat can't generate the same secrets generated by Alice and Bob.

After generating the six shared secrets, the pre-master secret is no longer needed and should, for security reasons, be deleted.

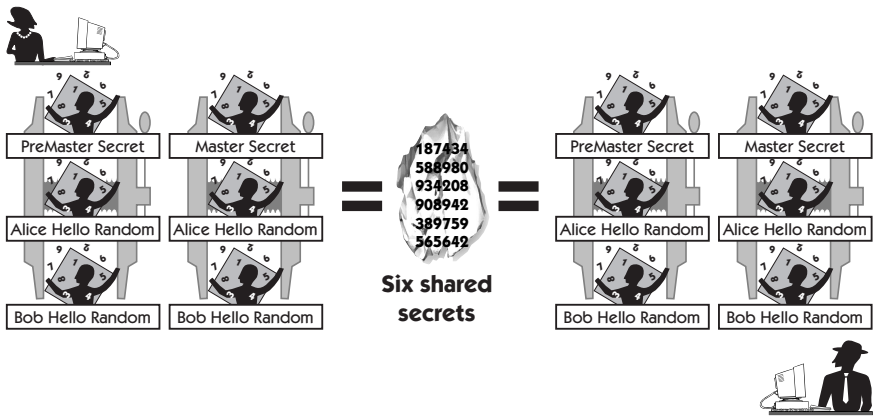

Figure 20-11 Alice and Bob use the pre-master secret, pseudo-random numbers, and pseudo-random functions to produce six shared secret values. Presumably, an attacker can't imitate their process.

Bob Authenticates Himself to AliceDotComStocks

If AliceDotComStocks wants to authenticate Bob, she requests his digital certificate after she sends hers to him. We'll pick up the action after Bob sends Alice the pre-master secret and his digital certificate containing his RSA public key.

Bob uses RSA to authenticate himself to Alice. Definition: certificate verify message

Bob authenticates himself to Alice by verifying that he has the private RSA key that matches the public RSA key on the digital certificate he just sent Alice. Bob digests a combination of the master secret and some previous messages. He then signs the digest (encrypts it with his private RSA key) and sends it to Alice. The message Bob sends to Alice (shown in Figure 20-12) is called a *certificate verify message*.

Alice verifies the certificate verify message with her copy of Bob's public RSA key. She trusts that she must be communicating with the owner of Bob's private key and the person who knows their shared master secret—Bob.

Message digest assurance in action

It's instructive and reinforcing to note that even though BlackHat can also decrypt this message digest with Bob's public key, he can't recover Alice and Bob's master secret because Bob digests the master secret before he signs it. Recall that a secure message digesting function, such as MD5 or SHA-1, implements one-way assurance (see Chapter 14). One-way assurance (and a sufficiently long master secret) prevents BlackHat from "going backward" to the master secret.

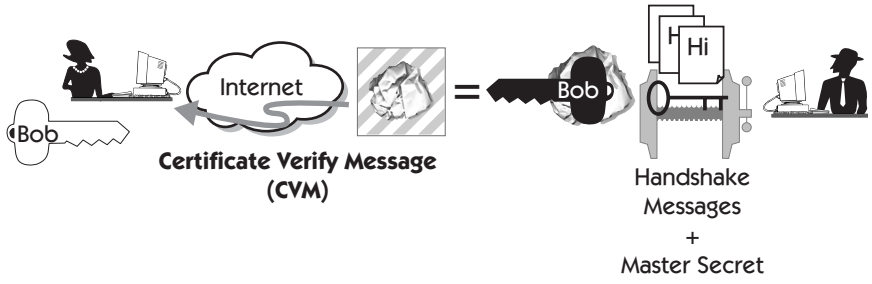

Figure 20-12 Bob authenticates himself to Alice. Not shown: Alice decrypts the CVM with Bob's public key and compares it to her independently hashed Handshake Messages and Master Secret.

Review

Secure Socket Layer (SSL) and the newer standard Transport Layer Security (TLS) are used to securely send information over the Internet, and the two standards are very similar. Both are designed to negotiate cryptographic parameters between two computers. SSL/TLS implements three cryptographic

assurances—authentication, confidentiality, and message integrity—and provides secure key exchange between an Internet browser and an Internet server.

SSL/TLS goes through the following steps to complete a secure transaction: negotiating parameters, exchanging digital certificates, secret key agreement (exchange), authentication, and bulk data encryption (confidentiality and message integrity).