"just do it" and introduce security functionality into their favorite applications. A classic example is Pretty Good Privacy (PGP), which provides secure e-mail (discussed later in this section). Requiring only client and server application code, PGP was one of the first security technologies to be broadly used in the Internet.

8.5.1 Secure E-Mail

We now use the cryptographic principles of Sections 8.2 through 8.3 to create a secure e-mail system. We create this high-level design in an incremental manner, at each step introducing new security services. When designing a secure e-mail system, let us keep in mind the racy example introduced in Section 8.1—the love affair between Alice and Bob. Imagine that Alice wants to send an e-mail message to Bob, and Trudy wants to intrude.

Before plowing ahead and designing a secure e-mail system for Alice and Bob, we should consider which security features would be most desirable for them. First and foremost is *confidentiality*. As discussed in Section 8.1, neither Alice nor Bob wants Trudy to read Alice's e-mail message. The second feature that Alice and Bob would most likely want to see in the secure e-mail system is *sender authentication*. In particular, when Bob receives the message "`I don't love you anymore. I never want to see you again. Formerly yours, Alice`," he would naturally want to be sure that the message came from Alice and not from Trudy. Another feature that the two lovers would appreciate is *message integrity*, that is, assurance that the message Alice sends is not modified while en route to Bob. Finally, the e-mail system should provide *receiver authentication*; that is, Alice wants to make sure that she is indeed sending the letter to Bob and not to someone else (for example, Trudy) who is impersonating Bob.

So let's begin by addressing the foremost concern, confidentiality. The most straightforward way to provide confidentiality is for Alice to encrypt the message with symmetric key technology (such as DES or AES) and for Bob to decrypt the message on receipt. As discussed in Section 8.2, if the symmetric key is long enough, and if only Alice and Bob have the key, then it is extremely difficult for anyone else (including Trudy) to read the message. Although this approach is straightforward, it has the fundamental difficulty that we discussed in Section 8.2—distributing a symmetric key so that only Alice and Bob have copies of it. So we naturally consider an alternative approach—public key cryptography (using, for example, RSA). In the public key approach, Bob makes his public key publicly available (e.g., in a public key server or on his personal Web page), Alice encrypts her message with Bob's public key, and she sends the encrypted message to Bob's e-mail address. When Bob receives the message, he simply decrypts it with his private key. Assuming that Alice knows for sure that the public key is

Bob's public key, this approach is an excellent means to provide the desired confidentiality. One problem, however, is that public key encryption is relatively inefficient, particularly for long messages.

To overcome the efficiency problem, let's make use of a session key (discussed in Section 8.2.2). In particular, Alice (1) selects a random symmetric session key, K_S, (2) encrypts her message, m, with the symmetric key, (3) encrypts the symmetric key with Bob's public key, K_B^+, (4) concatenates the encrypted message and the encrypted symmetric key to form a "package," and (5) sends the package to Bob's e-mail address. The steps are illustrated in Figure 8.19. (In this and the subsequent figures, the circled "+" represents concatenation and the circled "–" represents deconcatenation.) When Bob receives the package, he (1) uses his private key, K_B^-, to obtain the symmetric key, K_S, and (2) uses the symmetric key K_S to decrypt the message m.

Having designed a secure e-mail system that provides confidentiality, let's now design another system that provides both sender authentication and message integrity. We'll suppose, for the moment, that Alice and Bob are no longer concerned with confidentiality (they want to share their feelings with everyone!), and are concerned only about sender authentication and message integrity. To accomplish this task, we use digital signatures and message digests, as described in Section 8.3. Specifically, Alice (1) applies a hash function, H (for example, MD5), to her message, m, to obtain a message digest, (2) signs the result of the hash function with her private key, K_A^-, to create a digital signature, (3) concatenates the original (unencrypted) message with the signature to create a package, and (4) sends the package to Bob's e-mail address. When Bob receives the package, he (1) applies Alice's public key, K_A^+, to the signed message digest and (2) compares the result of this operation with his own hash, H, of the message. The steps are illustrated in

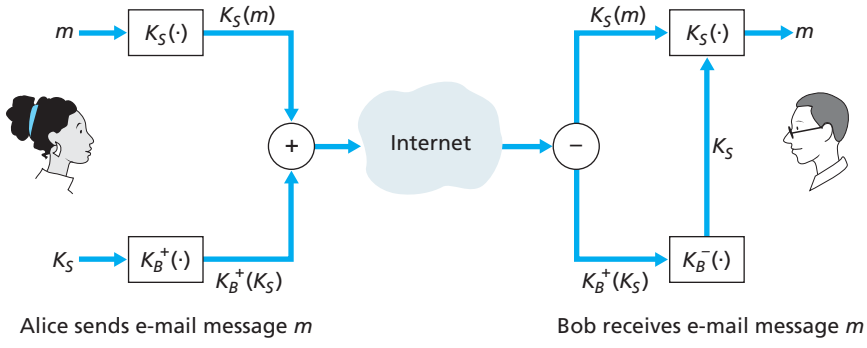

Figure 8.19 ♦ Alice used a symmetric session key, K_S, to send a secret e-mail to Bob

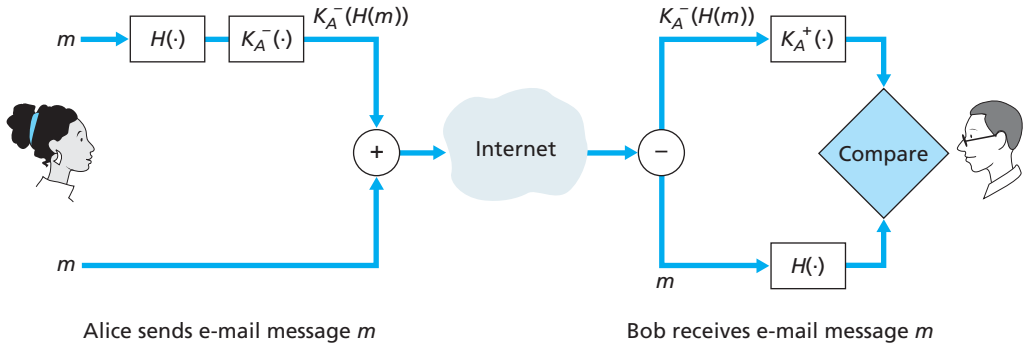

Figure 8.20 ♦ Using hash functions and digital signatures to provide sender authentication and message integrity

Figure 8.20. As discussed in Section 8.3, if the two results are the same, Bob can be pretty confident that the message came from Alice and is unaltered.

Now let's consider designing an e-mail system that provides confidentiality, sender authentication, *and* message integrity. This can be done by combining the procedures in Figures 8.19 and 8.20. Alice first creates a preliminary package, exactly as in Figure 8.20, that consists of her original message along with a digitally signed hash of the message. She then treats this preliminary package as a message in itself and sends this new message through the sender steps in Figure 8.19, creating a new package that is sent to Bob. The steps applied by Alice are shown in Figure 8.21. When Bob receives the package, he first applies his side of Figure 8.19 and then his side of Figure 8.20. It should be clear that this design achieves the goal of providing confidentiality, sender authentication, and message integrity. Note that, in this scheme, Alice uses public key cryptography twice: once with her own private key and once with Bob's public key. Similarly, Bob also uses public key cryptography twice—once with his private key and once with Alice's public key.

The secure e-mail design outlined in Figure 8.21 probably provides satisfactory security for most e-mail users for most occasions. But there is still one important issue that remains to be addressed. The design in Figure 8.21 requires Alice to obtain Bob's public key, and requires Bob to obtain Alice's public key. The distribution of these public keys is a nontrivial problem. For example, Trudy might masquerade as Bob and give Alice her own public key while saying that it is Bob's public key, enabling her to receive the message meant for Bob. As we learned in Section 8.3, a popular approach for securely distributing public keys is to *certify* the public keys using a CA.

CASE HISTORY

PHIL ZIMMERMANN AND PGP

Philip R. Zimmermann is the creator of Pretty Good Privacy (PGP). For that, he was the target of a three-year criminal investigation because the government held that US export restrictions for cryptographic software were violated when PGP spread all around the world following its 1991 publication as freeware. After releasing PGP as shareware, someone else put it on the Internet and foreign citizens downloaded it. Cryptography programs in the United States are classified as munitions under federal law and may not be exported.

Despite the lack of funding, the lack of any paid staff, and the lack of a company to stand behind it, and despite government interventions, PGP nonetheless became the most widely used e-mail encryption software in the world. Oddly enough, the US government may have inadvertently contributed to PGP's spread because of the Zimmermann case.

The US government dropped the case in early 1996. The announcement was met with celebration by Internet activists. The Zimmermann case had become the story of an innocent person fighting for his rights against the abuses of big government. The government's giving in was welcome news, in part because of the campaign for Internet censorship in Congress and the push by the FBI to allow increased government snooping.

After the government dropped its case, Zimmermann founded PGP Inc., which was acquired by Network Associates in December 1997. Zimmermann is now an independent consultant in matters cryptographic.

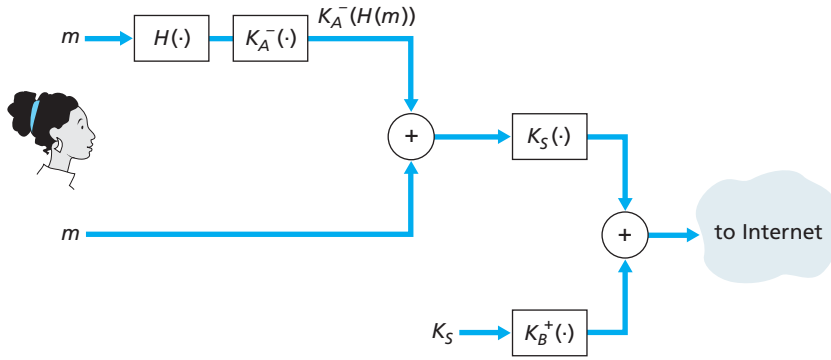

Figure 8.21 ♦ Alice uses symmetric key cyptography, public key cryptography, a hash function, and a digital signature to provide secrecy, sender authentication, and message integrity