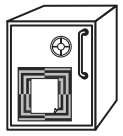
Chapter 7
SECRET KEY ASSURANCES

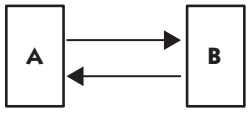
Definitions: confidentiality, authentication, integrity, nonrepudiation

Good cryptographic methods assure us that we can keep our secrets from others. That is, Alice and Bob's encrypted files remain private between them as long as their secret key stays secret.

Modern-day cryptographers use the term *confidentiality* to mean that your encrypted secrets aren't available to unauthorized users.[1] Let's review that concept briefly and examine three other necessary electronic data assurances—authentication, integrity, and nonrepudiation—defined in Figure 7-1.



Confidentiality is assurance that only owners of a shared secret key can decrypt a computer file that has been encrypted with the shared secret key.



Authentication is assurance of the identity of the person at the other end of the line. Authentication stops masquerading imposters.

Integrity is assurance that a file was not changed during transit and is also called **message authentication.**

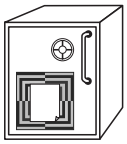
Nonrepudiation is assurance that the sender cannot deny a file was sent. This cannot be done with the secret key alone.

Figure 7-1 Terms of assurance.

1. In Chapters 7 and 8 we represent confidentiality with an image of a safe with an encrypted plaintext symbol. We're using a safe to reinforce the concept that encrypted text ensures privacy. After Chapter 8, confidentiality will be shown using only the encrypted plaintext symbol (without the safe).

In this chapter we explain how secret key cryptography implements these assurances. Later chapters examine how modern cryptography uses public keys more than secret keys for this purpose. The concepts are the same, although more involved, when public key methods are used. So we look first at the simpler case.

Confidentiality



Why you want authentication, integrity, nonrepudiation.

Suppose Alice and Bob have a West Coast real estate business. While Bob is on the road, Alice and Bob exchange financial and love notes encrypted with their secret key. Strong cryptography helps Alice and Bob feel assured their confidentiality (privacy) is being maintained because only someone who has their secret key can make sense of their shared electronic messages (see Figure 7-2).

Strong cryptography also ensures the confidentiality of encrypted files stored on computer disks; only those with whom we've shared the secret encrypting key can decrypt and understand the content.

But confidentiality (privacy) is not enough assurance to give you the warm fuzzies you crave about the security of your communications (see Figure 7-3). Even before you send or receive encrypted data to or from another computer, you need to know that the person on the other end of the line is the person he

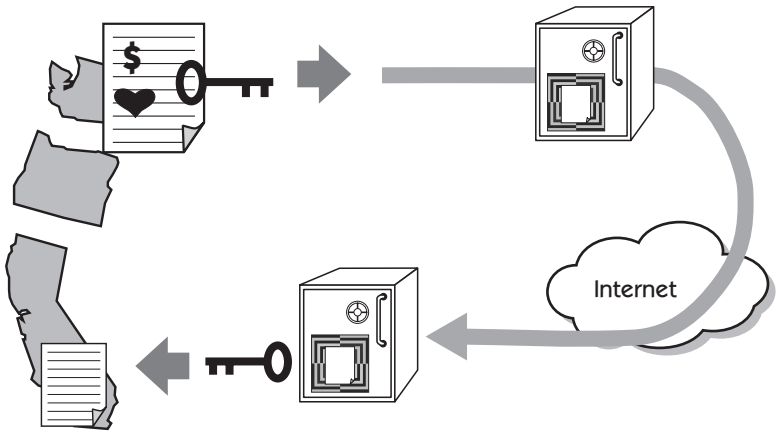
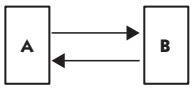


Figure 7-2 Confidentiality is like sending your secret in a safe; only the owner of the shared secret key can decrypt the message (open the safe).

Figure 7-3 Cryptography offers a way to detect masquerading impostors and ensure the identity of the person on the other end of the line.

or she claims to be (authentication). You also need to know that the software you downloaded hasn't been tampered with during its journey to you (integrity). And you'd probably also like to be assured that your stockbroker brother-in-law can't deny that he received your sell order before the bottom dropped out of the market. Similarly, he wants the same assurance if you deny that you instructed him to buy a falling-star dot-com (nonrepudiation).

Authentication



Enter BlackHat

Challenge and response: Alice authenticates Bob.

Shared secret keys can also be used to authenticate credentials. Cryptographic authentication assures Alice that her electronic contact is the genuine Bob and not someone masquerading as Bob—unless the masquerader has stolen a copy of Alice and Bob's shared secret key. Here's how Alice is assured it's the authentic Bob.

If Bob wants to send Alice information via computer, all Alice knows is that her phone rings, the modem picks up, and some computer requests access to her computer. Is it Bob? Or is it our book's bad guy, the nefarious BlackHat? How can Alice feel somewhat secure that it's Bob without seeing his face, hearing his voice, or asking for his mother's maiden name? She needs a genuine electronic ID from Bob.

If the computer requesting access to Alice's computer can verify that it knows Alice and Bob's shared secret key, Alice will feel more secure that it's Bob. But it would be foolish to ask Bob to send the secret key to prove it's his computer calling. BlackHat might be listening in and make a copy. Alice needs to know that the person on the other end of the line knows their secret key without either of them divulging it. A way to know whether the caller is the genuine, *authentic* Bob is called *challenge and response.* It's shown in Figure 7-4.

CHALLENGE AND RESPONSE

Authentication by Challenge Response

Alice sends a challenge.
She picks a number between 1 and 100, say 34, and challenges the computer requesting access to correctly encrypt 34. Only the secret key Alice shares with Bob will correctly encrypt 34.

Bob responds.
He encrypts 34. Say 34 encrypts to "%2." He sends **%2** back to Alice.

Alice finishes authenticating Bob.
She also encrypts 34 to %2 and is assured it's Bob. Only their shared secret key encrypts 34 to %2.

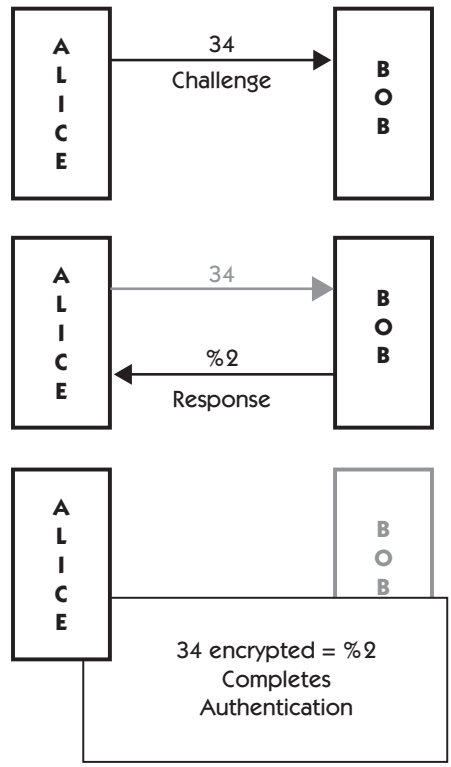


Figure 7-4 Alice authenticates Bob with a challenge and response protocol.

When the challenge and response protocol is completed, Alice is assured that the caller is Bob's computer. But note that Bob is not assured that the computer on the other end is Alice's because he has only responded to a challenge from someone he hopes is Alice. Bob must authenticate Alice, as shown in Figure 7-5.

Bob authenticates Alice.
Bob should challenge Alice in a similar way. Except Bob should challenge with any number other than 34. BlackHat may have listened in.

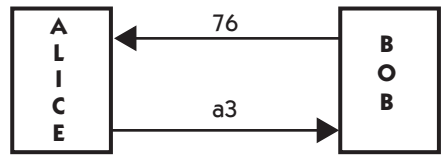
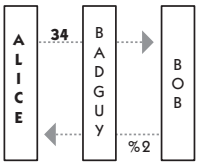


Figure 7-5 Bob authenticates Alice.



An Authentication Attack

Now suppose that BlackHat has listened in and recorded the challenge and response. Later, if Alice challenged with `34` again, BlackHat could impersonate Bob because he knows that Alice and Bob's shared secret key encrypts `34` to `%2`. That is, BlackHat intercepts Alice's challenge (e.g., 34) before it gets to Bob. He uses the previously recorded response (e.g., %2) and correctly responds to Alice.

Because Alice wants to ensure that she never again challenges Bob with the same number, she picks a random number from a very big group of numbers. A good cryptographic system chooses a challenge between 1 and a very, very big number. How big is big enough? Imagine all the sand on earth in a pile. Choose a grain of sand, put the grain back in the pile, mix up the pile, and choose a grain again. It's unlikely the same grain will be chosen twice. In fact, it's unlikely you could find the same grain again.

Good random numbers draw from a set of even bigger numbers so that you're even more unlikely to choose the same number again. We look at randomness next. You'll find more detail about randomness in Appendix A.[2]

Not Really Random Numbers

Do these numbers between 1 and 3 trillion seem random to you?

1414213562373

1732050807569

2236067977499

A good cryptographic system uses randomization with as little detectable pattern as possible. The lack of pattern is crucial for many reasons. Here's one. Say BlackHat records Alice's challenges to Bob. On Monday, Alice sends Bob the challenge `1,414,213,562,373;` on Tuesday, `1,732,050,807,569;` and on Wednesday, `2,236,067,977,499.` Bob correctly responds and is authenticated by Alice. The challenges may, at first glance, look like random numbers; but they actually follow a simple sequence. As shown in Figure 7-6, they

2. Although it's not shown here, passwords (and random values) should be long to prohibit BlackHat from successfully guessing them.

1.414213562373	*	1.414213562373	=	2
1.732050807569	*	1.732050807569	=	3
2.236067977499	*	2.236067977499	=	5

Figure 7-6 The square roots of 2, 3, and 5.

are the square roots of 2, 3, and 5, respectively (with decimal points removed). Although they look random, they're not because it's easy to figure out the next number in the sequence.[3]

BlackHat successfully masquerades as Bob to Alice.

If BlackHat figures out Alice's sequence, he can impersonate Bob. BlackHat guesses that Alice's next challenge will be the square root of 6 (2.449489742783) with the decimal point removed. BlackHat knows he can't correctly respond to Alice's challenge of `2,449,489,742,783` because he doesn't have the correct secret key. But he may be able to trick Bob into doing the work for him. BlackHat intercepts Bob's next call to Alice's computer. Bob thinks he's connected to Alice. BlackHat challenges Bob with `2,449,489,742,783`. Bob encrypts the challenge and responds to BlackHat. BlackHat now knows how to respond to Alice's challenge of `2,449,489,742,783`. He puts Bob on hold while he calls Alice's computer. BlackHat logs on to Alice's computer and drops Bob's connection. When Bob tries Alice's computer again, he gets a busy signal.

Computer cryptography crucially relies on random numbers. But almost the most difficult task you can give a computer is to make something random. Even though computers are made to behave in the same identical way over and over again, many, if not most, people think computer work is already frustrating enough. Imagine if a computer behaved differently on different days (ugh).

One-time pad

Making Good Use of Randomization

One cryptographic system that absolutely cannot be cryptanalyzed wasn't conceived of until the twentieth century. In this system—termed the *one-time pad*—a randomly generated, nonrepeating key (the length of the key is at least as great as the length of the message) is used only once. The one-time use ensures that ciphertext can't be cryptanalyzed through an examination of patterns in different messages. Perfect secrecy is ensured only by a truly random number. Quantum events, such as those measured by a Geiger counter, are believed to be the only source of truly random information.

(Continued)

3. See Appendix A for more on pseudo-random numbers.

The one-time pad got its name from Germany's use of this system around the 1920s. The Germans typed a sequence of supposedly random numbers on two separate pads—one for the receiver and one for the sender—to be used only once. The German system had a mechanical precursor, called the one-time system, that was developed independently. It was created by AT&T engineer Gilbert Vernam, who was studying security problems with the teletypewriter. It was improved on by U.S. Army Major Joseph Mauborgne, who proposed modifying Vernam's system by using a nonrepeating random key.

Definition: pseudo-random

In fact, computer programs can't make random numbers. They may come close, but not close enough to be called random. Instead, "random" numbers made by computer programs are actually *pseudo-random*. To you and me, a pseudo-random number may look like a random number, and we can use it as though it were a random number. Economists, statisticians, scientists, and others use pseudo-random numbers all the time. Nevertheless, if a cryptographer isn't very careful in using pseudo-randomness, a hound-dog cryptanalyst might spot it and use it to launch a successful attack.

Integrity

Did you just download the latest version of your favorite browser, or maybe a virus update program, from Microsoft or Netscape? You may have logged on through your network or dialed into an Internet service provider (ISP). Then you downloaded the file. The file passed from the vendor's disk to its Internet server to the Internet cloud to your ISP and finally to your machine (see Figure 7-7).

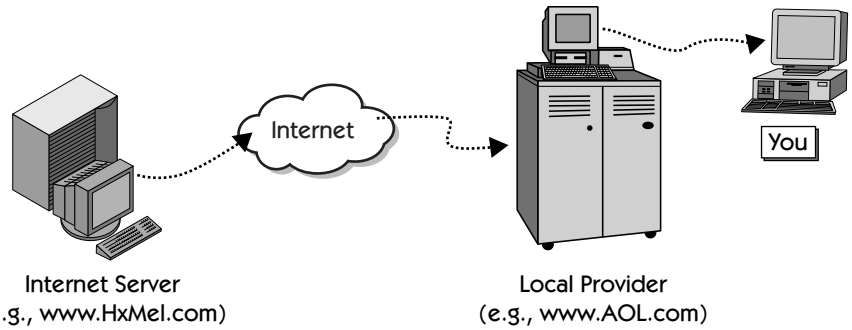


Figure 7-7 When you send or receive a message over the Internet, it's important to ensure its integrity.

You hope you received what you ordered; you hope that during each stop and forward along the way no one changed the contents. No one wants BlackHat to modify a virus checker program so that it will fail to check for particular viruses.

Integrity is often referred to as message authentication.

Authentication assures Alice that no one is masquerading as Bob. The principle of integrity assures Alice that no one can change Bob's messages without being detected. Authentication and integrity are very closely linked; integrity is often referred to as message authentication.

Definition: MAC

Alice and Bob obtain integrity assurance by using their secret key and the message to make a message fingerprint, known as a *message authentication code* (MAC). The message and message fingerprint are a closely tied, matched pair. No one can easily find another message that makes the identical matched message fingerprint. As with secret key encryption, the secret key Alice shares with Bob ensures that the message fingerprint is secure from forgeries. Even searching for an identical fingerprint is frustrating because the message fingerprint formula is designed so that a change in a single message letter (actually a single bit) changes about one-half the message fingerprint (see Figure 7-8).

Any change in the message changes the fingerprint. If `$10.00` is changed to `$10.01`, it makes a completely different message authentication code. As discussed in Chapter 5, this is called the avalanche effect and means that cryptanalysts have a difficult time knowing when they're close to producing a successful forgery.

Like a secret key encryption method, the MAC formula is publicly available and known; it's not secret. Chapters 13 and 14 cover this topic in greater depth.

Using the MAC for Message Integrity Assurance

In Figure 7-9, Alice creates the MAC from the secret key and the message `$10.00 is cost of game` and sends both the message and the MAC to Bob.

Figure 7-8 A slight difference between two messages results in greatly differing message authentication codes.

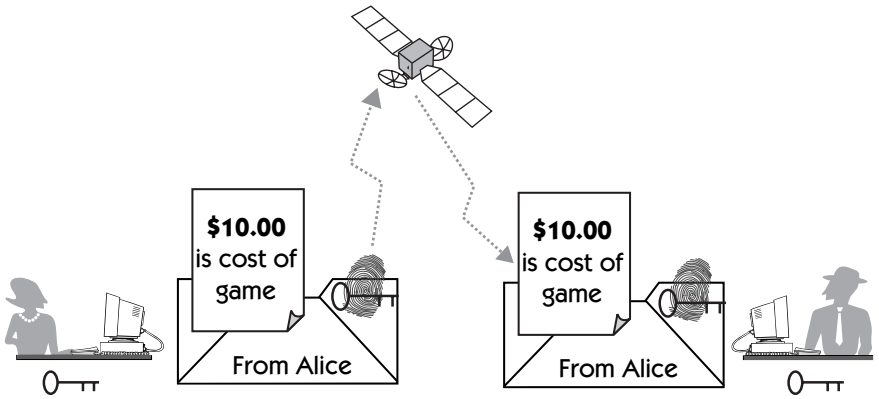


Figure 7-9 Alice makes a message and a MAC and sends both to Bob.

In Figure 7-10, Bob uses the message and his copy of their shared secret key to independently calculate another message fingerprint. If Bob's independently calculated message fingerprint is exactly equal to the message fingerprint he received from Alice, he is assured that the message has not been changed in transit.

MAC formula and shared secret key ensure that no one else can duplicate the (MAC) message fingerprint.

Bob can feel secure because no one else can duplicate the message fingerprint without knowing the secret key he shares with Alice. Note that in our example, the message `$10.00 is cost of game` is sent as plaintext; that is, the message is not confidential. In Part IV, "Real World Systems," we'll see how to get confidentiality, authentication, and integrity assurances together.

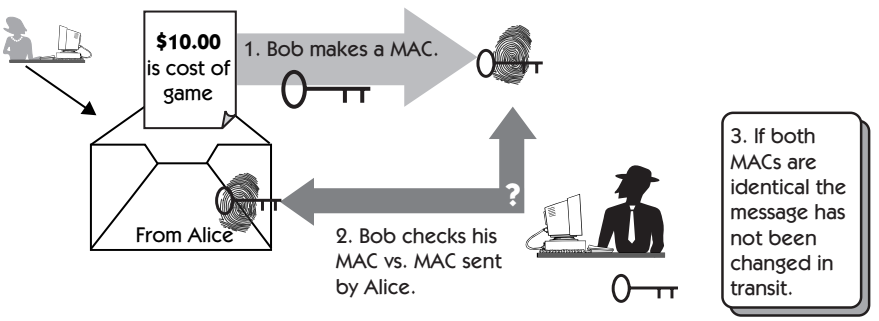


Figure 7-10 Bob verifies that Alice's message arrived unaltered.

Why Bother Using a Message Authentication Code?

A MAC is small.

Why bother computing a MAC if you're going to send the message in the clear anyway? First, software updates can be many megabytes and don't need to be secret. Encryption and decryption take computer time. A MAC is very small compared with the size of most messages.

Confidentiality, by itself, does *not* ensure integrity.

Second, it's a common but mistaken notion that secret key confidentiality alone assures message integrity. Encryption doesn't stop BlackHat from altering messages (see Chapter 22). That is, message fingerprints are as important as confidentiality and should always be sent with the messages.

File and MAC Compression

By the way, MAC math compresses large files to a very few bytes (characters), but you shouldn't confuse MAC compression with popular compression programs such as PKZip, WinZip, or StuffIt. MAC compression is one-way; there's no way to decompress a MAC to reclaim the original underlying text (see Figure 7-11).

In fact, if BlackHat could recover or figure out any part of the original text from a MAC, the MAC program would be flawed and insecure (see Chapter 14).

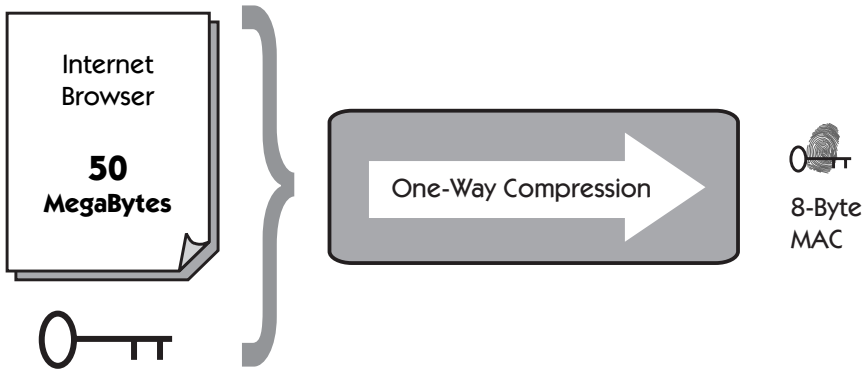


Figure 7-11 MAC compression is one-way. The MAC cannot be decompressed to recover the original file (message).

Nonrepudiation: Secret Keys Can't Do It

In a world of marital harmony and perfect people, Alice would never deny or forget that she received a message from Bob, and vice versa. But suppose Alice and Bob share a secret key with their stockbroker. Alice or Bob encrypts a buy order, specifies a price, and sends it to Untrusty the stockbroker. Untrusty thinks the price will go down and decides not to buy Alice and Bob's stock until the next day; Untrusty figures he can pocket the difference.

But the next day the price goes up, and Untrusty claims he never received the order from Alice or Bob. Untrusty denies, or *repudiates*, the buy order. Bob and Alice would be out of luck except that cryptography provides a way to make Untrusty confess the truth.

Alice makes Untrusty agree to encrypt a message stating that he, Untrusty, received their buy order. Then if Untrusty denies or repudiates the buy order, Alice could show the message to an impartial judge. Alice could say, "Here is the encrypted message and the decrypted message that Untrusty sent us; here is the secret key Untrusty used to make the encrypted message. Only the secret key, which is shared by Untrusty and us, can make this exact encrypted message from the plaintext. It's proof that Untrusty sent us the message that he received our buy order."

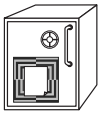
Untrusty's lawyer slowly gets up, slowly straightens his tie, and very slowly touches his finger to his lip. He's getting paid by the hour. Untrusty's lawyer asks Alice whether she has a copy of the secret key she shares with Untrusty. Alice, of course, has a copy. How else could she decrypt the encrypted confirmation she received from Untrusty? The lawyer then asks Alice whether she could also make exactly the same message and encrypt it so that it looks exactly like what she says was sent to her by Untrusty. She can. Alas—this means that she has no case.

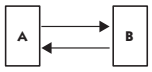
Secret keys alone aren't enough to ensure that someone else can't repudiate or deny receiving your message. To implement nonrepudiation, you must either use secret keys with a trusted third party (a process reviewed in Chapter 8) or use public key cryptography.

Alice and Bob—along with their offspring, Casey and Dawn—will encounter another difficulty with secret key cryptography, a problem resolved by public key cryptography. They are computer-savvy and know to encrypt their messages with Triple DES or the new AES standard, Rijndael. If they are to communicate securely with one another and the diverse others they meet at the four corners of the Internet world, they'll need to exchange and keep track of many secret keys. Although they are all keyed up to share secrets globally, Chapter 8 shows the problems of a secret-key-only system.

Review

Cryptography and shared secret keys can be used to secure electronic files and communications. Cryptographic assurances are categorized as follows.





- Confidentiality is assurance that only owners of a shared secret key can decrypt a computer file that has been encrypted with the identical shared secret key.
- Authentication is assurance of the identity of the person at the other end of the line. Because Bob can't send the shared secret, Alice challenges Bob to correctly encrypt a previously unused random number with their shared secret key. Only the shared secret key will correctly encrypt the random number.
- Integrity, or message authentication, is assurance that a file has not been changed during transit. A message and a shared secret key make a unique message authentication code (MAC), or message fingerprint. Only someone with a copy of the shared secret key can correctly reproduce the fingerprint.
- Nonrepudiation is assurance that the sender cannot deny that a file was sent. This cannot be done using a secret key alone; it requires a mutually trusted third party or public key technology.