Chapter 14

Message Digest Assurances

Chapter 13 ended with BlackHat trying to forge a newsletter whose message digest exactly equaled Alice's newsletter digest. This chapter shows how cryptographers stop that kind of attack (see Figure 14-1) and others by using message digest security features. You'll see how message digest design stops BlackHat from making undetectable modifications to Alice and Bob's messages.

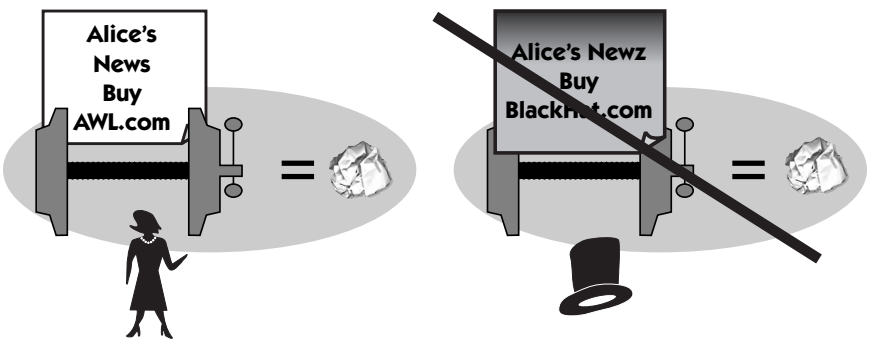

Figure 14-1 BlackHat can't find a newsletter that makes the same message digest as Alice's newsletter. This means that he can't fool Bob into accepting his forged newsletter.

Two Message Digest Flavors

Message digests come in two flavors—keyed and non-keyed—as shown in Figure 14-2.

Definition: modification detection codes

Non-keyed message digests are made without a secret key and are called message integrity codes (MICs) or *modification detection codes* (MDCs). MIC is more commonly used, but MDC seems to be a more straightforward description of how a non-keyed message digest works. Most public key digital signatures use non-keyed message digests.

Definition: message authentication codes

Keyed[1] message digests, known as *message authentication codes* (MACs), combine a message and a secret key. MACs require the sender and receiver to share a secret key. This chapter reviews and extends the discussion of MACs presented in Chapter 7, where you saw the shared secret key used in conjunction with a secret key encryption method, such as DES or Rijndael (the new AES standard).

Note that although the term *hash function* is usually reserved for non-keyed message digests, it is sometimes used to refer to both keyed and non-keyed digest functions.

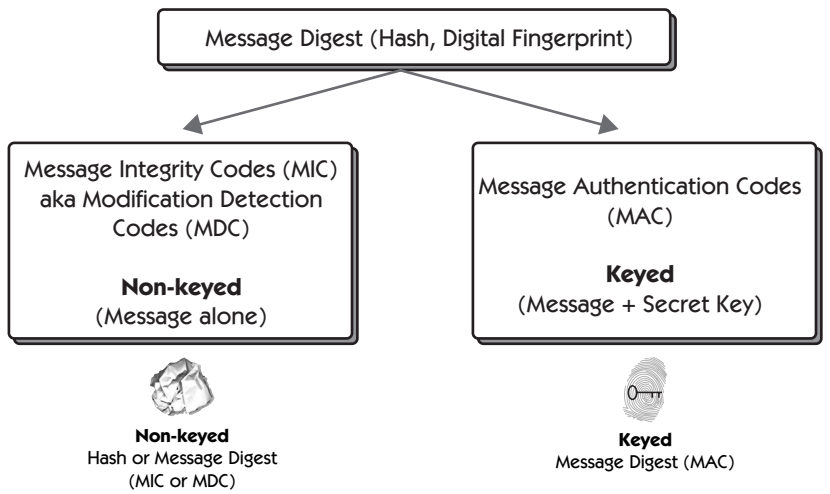

Figure 14-2 Two types of message digests, keyed and non-keyed.

1. "Keyed" does not mean that the message digest is signed (private key encrypted). Instead, it means that the digest is made with a secret key.

Non-keyed Message Digest Assurances

Non-keyed message digest assurances: one-wayness, weak collision, strong collision

Non-keyed message digests implement three security assurances—one-wayness, weak collision resistance, and strong collision resistance—to stop BlackHat from making message digest forgeries. We'll describe and show some examples of each of these security assurances.

One-wayness

One-wayness ensures that you can't recover the newsletter from a digest. The digest process is irreversible. If you are trying to find the person who made a particular fingerprint without matching the fingerprint against a comprehensive databank of fingerprints and their owners, it's not likely that you'll succeed. Similarly, after Alice hashes her newsletter, it's computationally infeasible to reverse the process; that is, given the newsletter digest, you can't find a newsletter that made it, as shown in Figure 14-3.

Because one-wayness ensures that the original message is not recoverable, a message digest doesn't have to be sent (or stored) as a confidential message.

Collision Resistance

Message digest collision resistance has much in common with airline seating. We'll use the airline seating analogy to describe both weak and strong collision resistance.

Figure 14-3 One-wayness: message digest computations are irreversible.

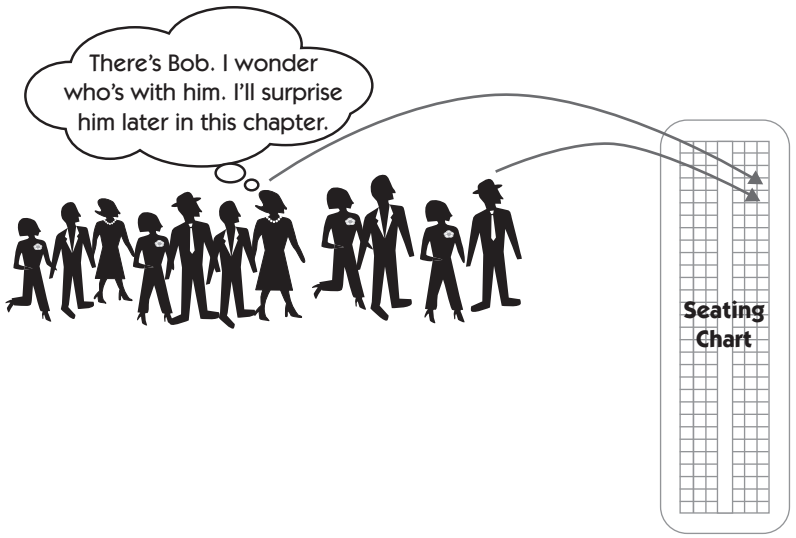

Figure 14-4 The Cryptographic Airlines seat reservation program is collision-resistant. Each passenger is assigned a particular seat; no two passengers are ever assigned the same seat.

Suppose that Cryptographic Airlines (CryptoAir) has a seat reservation program that assures passengers and CryptoAir's gate personnel that each passenger is assigned a particular seat and that no two passengers are ever assigned the same seat (see Figure 14-4). CryptoAir's seating program is collision-resistant.[2]

Weak Collision Resistance

Weak collision resistance prevents BlackHat from perpetrating the fraud shown at the end of Chapter 13.

Let's say Alice reserves seat 3A on a CryptoAir flight (shown in Figure 14-5). Weak collision resistance assures her that BlackHat can't convince CryptoAir to assign seat 3A to any other passenger. Similarly, weak collision-resistant hash functions assure Alice that BlackHat can't find another newsletter whose digest is exactly equal to Alice's newsletter digest.

Weak collision resistance stops BlackHat from successfully convincing Bob to accept his fraudulent newsletter. When Bob independently calculates a newsletter digest from BlackHat's newsletter, it does not equal the decrypted newsletter digest Alice made and sent (see Figure 14-6).

2. Of course, CryptoAir's pilots are also, hopefully, collision-resistant!

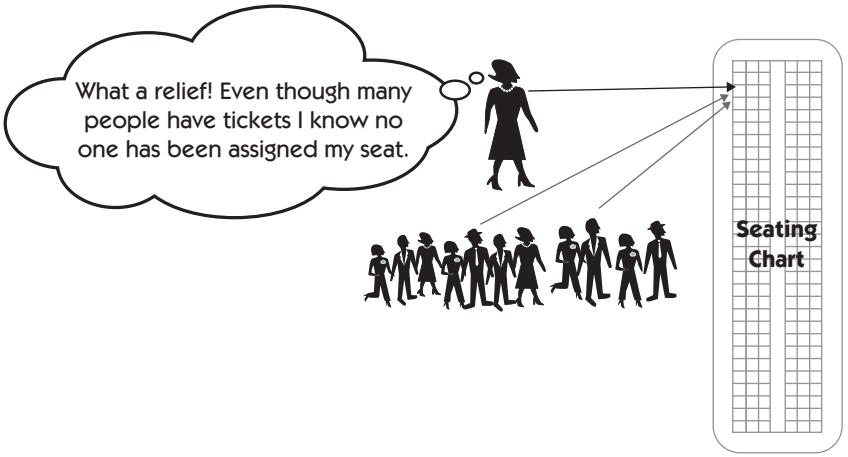

Figure 14-5 Alice is assured that no other passenger is assigned her seat, 3A.

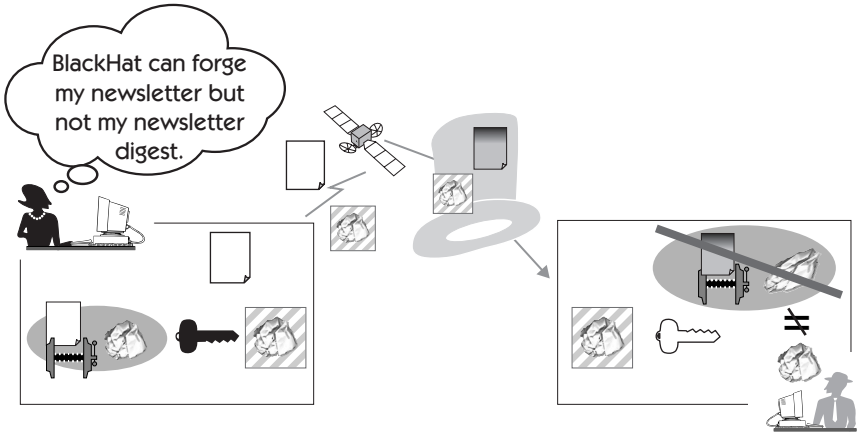

Figure 14-6 Weak collision resistance assures Alice and Bob that BlackHat's forged newsletter will not hash into one identical to Alice's newsletter digest.

Examples of One-way and Weak Collision Resistance

Following are two problems solved by one-wayness and weak collision resistance: detecting modifications and sharing data without revealing contents.

Detecting Modifications

Using one-wayness and weak collision resistance to detect modified files

Suppose Alice wants to know whether any file on her computer is modified while she's gone. She can't trust file dates because they're too easily modified.

A poor solution is for Alice to copy all the files to a backup device and carry it with her everywhere. Not only is this approach inconvenient, but also if the backup copy is lost or stolen, her files are at risk.

A better solution is for Alice to make a message digest and write the 160-bit output (about 20 characters) on a piece of paper. When she returns, she re-runs the digest program and checks the new message digest output against the message digest she took with her. If they're equivalent, she's guaranteed that not even one bit, of any file, was modified while she was gone.

One-wayness and weak collision resistance help Alice protect her data. One-wayness ensures that the message digest does not give any clue as to the contents of the files because it's impossible to get back to the original documents after they have been hashed (digested). Weak collision resistance ensures that BlackHat can't modify any file in such a way that the modified file will result in an equivalent message digest.

Cryptographers have other, even more cryptic names for these two hash assurances. They refer to one-wayness as first pre-image resistance and to weak collision resistance as second pre-image resistance.

Sharing Data Without Revealing Contents

When Alice and Bob were married, Alice asked for a prenuptial agreement (she had made a bundle in her dot-com brokerage firm). Now she wants an attorney to hold and be able to verify the agreement, but she doesn't want the attorney to know what's in it unless she and Bob divorce.

Using one-wayness to share a secret without revealing its contents

To implement her plan, Alice hashes the prenuptial agreement and sends a copy to their attorney. The hashed prenuptial is a proxy for the plaintext prenuptial, but the attorney has no feasible way of knowing what is in the plaintext prenuptial. If Alice and Bob call it quits and someone disputes the contents of the prenuptial, Alice can send the attorney a copy of the plaintext prenuptial (along with a substantial retainer). The attorney hashes the plaintext and compares it to the message digest Alice sent him originally. If they're equivalent, it means that the plaintext prenuptial has not been modified since the message digest was made.

Weak collision resistance prevents Bob from making a forgery.

One-wayness prevents the attorney, or anyone else, from recovering the prenuptial from the digest. Weak collision resistance, in turn, prevents Bob from creating a forged prenuptial that hashes into a digest equivalent to the one held by the attorney. If Bob could find another prenuptial that would hash into an equivalent digest, the attorney would have no way of knowing which prenuptial was the original and which was the forgery.

In most instances one-wayness and weak collision resistance offer enough cryptographic assurances to prevent forgeries. Nevertheless, sometimes weak collision resistance doesn't have sufficiently formidable barriers. In those cases, strong collision resistance is needed. Let's see how it works.

Strong Collision Resistance

Cryptographic Airlines' gate personnel want assurance that no two passengers are assigned the same seat (see Figure 14-7). Similarly, no two messages can be found that make equivalent digests.

Definition: strong collision resistance

Whereas Alice is concerned only about herself and seat 3A (her reserved seat)—something that is ensured by weak collision resistance—CryptoAir's gate personnel are concerned with every passenger and every seat.[3] For that, they need strong collision resistance, or the assurance that no two passengers are ever assigned the same seat. The difference is a subtle but important one. Weak collision resistance stops forgery of a particular message; *strong collision resistance* stops forgery of any message.

Let's illustrate the difference between weak and strong collision resistance by taking a second look at Alice and Bob's prenuptial agreement.

Although weak collision resistance stops Bob from forging a prenuptial that hashes to the prenuptial Alice created, it doesn't stop Alice from creating such a forgery. How can this be? Suppose that Alice and Bob call it quits, and Alice decides to cut him off from as much money as possible. Bob claims that the prenuptial agreement contained sentences entitling him to 50 percent of all the

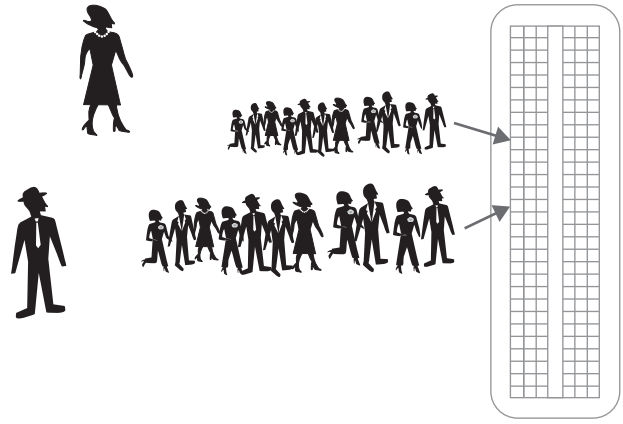

Figure 14-7 Strong collision resistance assures each and every passenger that his or her seat won't be given to another passenger.

3. CryptoAir never overbooks its flights and is, of course, an imaginary airline.

money earned during the marriage; the agreement is shown on the left side of Figure 14-8. Bob sighs with relief when the prenuptial hashes to the value given to Alice's attorney many years ago. But then Alice produces another prenuptial, and it contains the sentence shown on the right side of Figure 14-8. To Bob's amazement, it also hashes to the same value. How did Alice defeat weak collision resistance?

Recall that weak collision resistance stops Bob or Alice (or anyone) from making another prenuptial that hashes to an equivalent digest to the one shown on the left side of Figure 14-8, the digest Alice and Bob sent their attorney. But because Alice created the original prenuptial, she gave herself a much easier task. Here's what she did. When Alice created the prenuptial agreement, she made many prenuptials, each of which effectively states, "Bob gets one-half," shown on the left side of Figure 14-9. She then created many prenuptials that cut Bob off, shown on the right side of Figure 14-9. She had only to match *any* one agreement from the left side to *any* one from the right side.

Bob must match the particular prenuptial Alice gave him. Because Alice can create many prenuptials before she selects a particular prenuptial to give to Bob, she can match any one of them.

Birthday attack

The scenario just described is often called a *birthday attack* because it uses the analogy of finding equal birthdays. Suppose that Bob's birthday is February 1. If Bob wants to find another person who shares his birthday, he'll have to ask about 180 people before he likely finds someone whose birthday is also February 1. But Alice's task is much easier; she need only find any two people who have the same birthday. Let's say the first three people she asks have the birthdays February 1, June 2, and September 3, respectively. If the fourth person she asks shares any one of those birthdays, Alice's search is finished. If the fourth person's birthday is December 10, the fifth person Alice asks need only share any of the previous four birthdays. Alice's task takes much less effort (and time) than Bob's task.

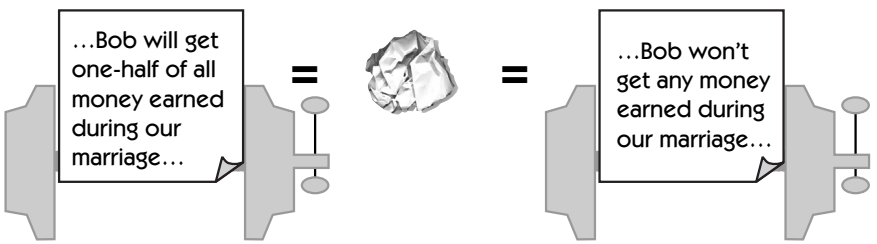

Figure 14-8 Bob and Alice have different prenuptials that hash to the same digest. Alice gave Bob their prenuptial when they were married. How did she find another prenuptial that hashed to the same digest as the one she gave Bob?

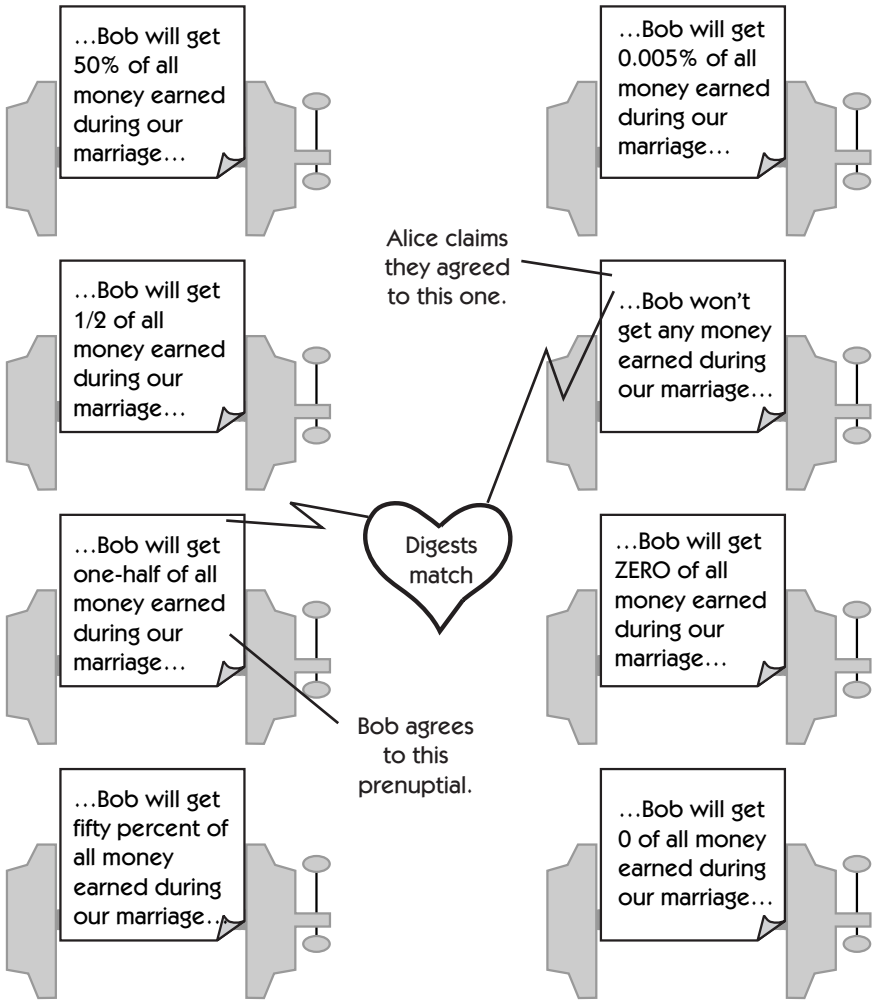

Figure 14-9 Alice makes many potential prenuptials. By the way, note that the last two messages in the right-hand column make different message digests; `ZERO` digests to a different value than `0` does.

Fortunately, cryptographers have compensated for this possibility, making message digests about a trillion trillion times larger (2^{80}). In other words, although Alice has a much easier forgery task than Bob, her task is still so time-consuming that it's infeasible.

If you're given a document, make a small change before signing it.

Although strong collision resistance prevents these kinds of attacks, there is an important lesson to be learned. Before Bob signs the prenuptial he should make a small and inconsequential change to the agreement Alice gave him. The small change ensures that none of Alice's potential forgeries, shown in the right-

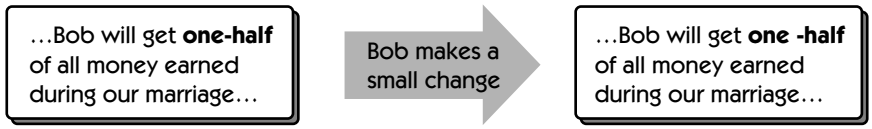

Figure 14-10 Bob adds a blank space after the word *one* and eliminates Alice's ability to use coincidental collisions such as those shown in Figure 14-9.

hand column of Figure 14-9, will hash to the identical digest that Bob signed. As shown in Figure 14-10, even adding a space after the word *one* will change the message digest enough so that it's unlikely to match any forgery Alice might make.

Non-keyed Digest Implementations

The two most popular non-keyed message digest programs are Message Digest 5 (MD5) and Secure Hash Algorithm-1 (SHA-1). Ron Rivest, the *R* in RSA, invented MD2 soon after he worked on RSA. MD5 is the latest version. NIST (the National Institute of Standards and Technology) and NSA enhanced MD4, MD5's predecessor, to produce SHA-1.

MD5 (128 bits) is more susceptible to birthday attacks than SHA-1 (160 bits).

MD5 makes a 128-bit digest; this means that a birthday attack (an attack against MD5's strong collision resistance) can be mounted using 64 bits (128/2), and that's now considered too vulnerable; MD5 collisions have been found for particular kinds of messages. (Recall that DES is considered vulnerable because it uses a 56-bit secret key.) Let's see why SHA-1 and other message digest programs are more secure against birthday attacks.

SHA-1 (160 bits): the popular current choice

SHA-1, often abbreviated to SHA, makes a 160-bit digest; this means that a birthday attack can be mounted using 80 bits (160/2). Mounting an 80-bit birthday attack requires about 65,000 times more effort than mounting a 64-bit attack.[4] This means that a successful one-hour attack against MD5 would need many years (65,000 hours) to be successful against SHA-1. So far, SHA-1 has been immune to cryptanalytic attacks successfully mounted against MD5.

Although not as widely known as MD5 and SHA-1, RIPEMD-160 is another secure 160-bit non-keyed digest. In fact, Hans Dobbertin, the cryptanalyst who successfully attacked MD5, was also a cryptographer on the team that developed RIPEMD-160. Like MD5 and SHA-1, it's also based in large part on MD4.

4. $2^{80}/2^{64} = 2^{16} = 65{,}536$.

Keyed Message Digest Assurances

Whereas an MDC depends on one-wayness and collision resistance to ensure security, message authentication codes (MACs), the other type of message digest, bases its security on using secret keys.

MAC has two parts: message and secret key.

Let's review and expand the example of MAC shown in Chapter 7. No matter what type of MAC you use, they all have two parts: a message (or file) and a secret key. A MAC combines and then compresses a message (e.g., `Internet Browser`) and a shared secret key (see Figure 14-11). Many, if not most, MACs are made with secret key block ciphers, such as DES. As with MDCs, the MAC file acts as a proxy for a much larger file. In our example, Bob verifies the authenticity of the sender (Alice) and the message `(Internet Browser)` by comparing the MAC Alice sent him against the MAC he independently produces.

The secret key cipher (e.g., DES) and shared secret key make BlackHat's forgery attempts computationally infeasible. This means that MAC security is closely tied to the security of the underlying secret key cipher and secret key.

A MAC Made with DES

The Data Authentication Algorithm (DAA) is made using DES and a compression method. It's also the subject of the Federal Information Processing Standards (FIPS-113) publication "Computer Data Authentication," which was published in 1985 by NIST.

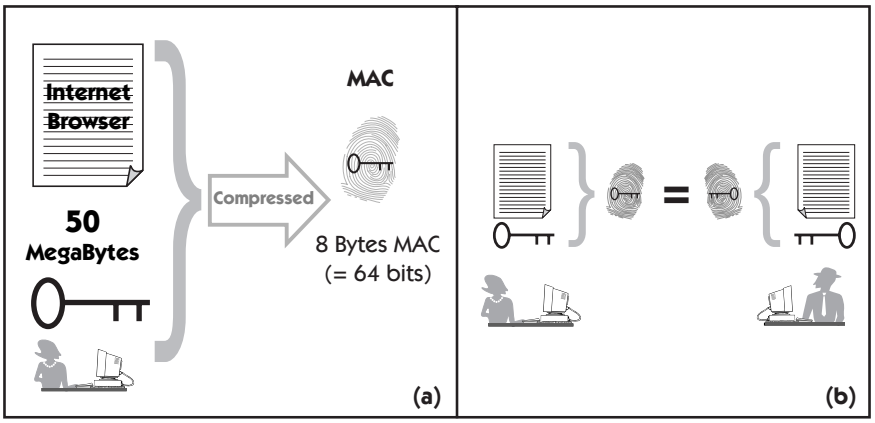

Figure 14-11 (a) A MAC is a digital fingerprint of a message and a secret key. (b) Using their shared secret key, Bob independently creates the identical MAC of the message sent by Alice.

Spring 2000: American Bankers Association removes DES-MACs from recommended list.

Because DES keys (56 bits) are believed to be increasingly susceptible to brute force cryptanalysis, DES-MACs are similarly susceptible. Although (as of spring 2000) NIST hasn't withdrawn DES-MACs as a U.S. federal processing standard, a prominent banking standards committee (American Bankers Association) removed DES-MACs from the recommended list, advising readers, "Continue to use single DES-based X9.9 [MACs] until a replacement is implemented . . . [with] actions that can be taken to reduce the risks." Its recommended actions include using triple-DES-like MACs, public key cryptographic methods, and modification detection codes (MDC), discussed earlier in this chapter.

A widely used MAC will most likely evolve from Rijndael, the new Advanced Encryption Standard. The following explanations use DES-MACs because the new MAC will probably behave like a DES-MAC.

DES-MAC Security

The two possible brute force attacks against MACs attack either the secret key or the MAC itself.

Definition: selective MAC forgeries

Because the DES and DAA methods are known, if the secret key isn't secret, anything done with DES/DAA is insecure. If BlackHat sniffs the line and gets a valid plaintext/MAC pair, he tries all the possible keys, as shown in Figure 14-12. At least one key must make the correct transformation.[5] After BlackHat gets the secret key, he can manufacture a MAC for any selected plaintext, a potentially damaging forgery often referred to as a *selective forgery*.

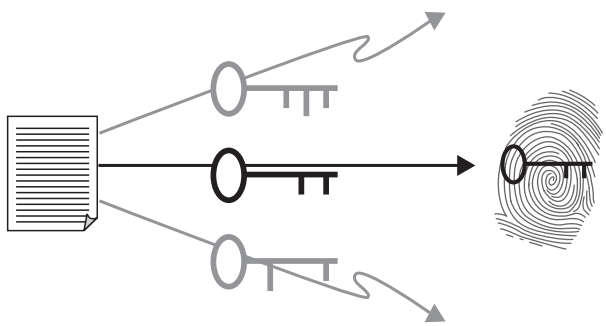

Figure 14-12 BlackHat figures out the secret key used to make the MAC.

5. Although it's not important to our discussion, it's possible that more than one key can make the matching MAC; in that case, BlackHat needs more plaintext/MAC pairs to determine the correct key.

Definition: existential MAC forgery

Even without knowing the secret key, BlackHat is also successful if he can forge any valid plaintext/MAC pair. In this case, referred to as an *existential* forgery, BlackHat can't choose a particular plaintext message. Instead, he's simply trying to find a pair that will verify when properly digested with the valid secret key.

For example, in Figure 14-13 BlackHat tries different plaintexts until one of his plaintext digests matches the digest being attacked. Existential forgeries need online verification, but that's not out of the question with automated response systems. Because BlackHat must try many different variations of plaintext, this attack is usually less damaging than a selective forgery. But even existential forgeries can be damaging.

Example: A damaging existential forgery

Suppose BlackHat has earned a $10 rebate and knows that the vendor is sending the rebate via an electronic check and associated MAC to his bank. He sees this as a perfect opportunity to increase the amount of the check. Although BlackHat lacks the secret key and thus can't control the exact amount of his forgery, he can try all numbers between 1 billion and 9 billion (8,000,000,000 numbers). If the MAC is only 32 bits long, he's guaranteed a match after about 4 billion[6] tries.

The lesson is this: It's probably advisable to use much bigger MACs, no less than 128 bits[7] and arguably no less than 160 bits.

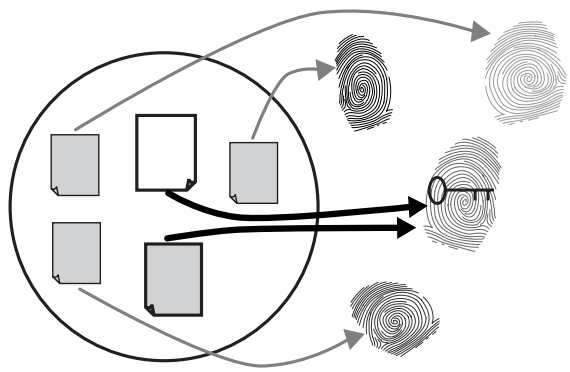

Figure 14-13 BlackHat finds another message that makes the same MAC as the white message; the other messages don't make an identical MAC as the white message.

6. 32 bits can hold only about 4 billion (4,000,000,000) different combinations.
7. 128 bits can hold about 4 billion x 4 billion x 4 billion x 4 billion different combinations.

Message Digest Compression

As mentioned in previous chapters, cryptographic compression is similar to that of popular compression programs such as PKZip, WinZip, and StuffIt, with the difference that cryptographic compression is one-way. Plaintext cannot be recovered after it is cryptographically compressed. MDCs and MACs compress files in about the same way except that MACs add the secret key to the cauldron.

Figure 14-14 shows how one particular MAC (DAA) compresses a message (or file) of any length (smaller than about one million trillion characters) to 64 bits.[8] The message is broken into 64-bit (8-byte) chunks. The first two 64-bit chunks are compressed together, and then DES encryption is applied to the new 64-bit chunk. The new chunk is then combined with the third 64-bit chunk, DES encryption is applied to it, and so on. So whereas a DES plaintext encryption produces ciphertext about the same size as the original plaintext, a MAC is much smaller than the plaintext file.

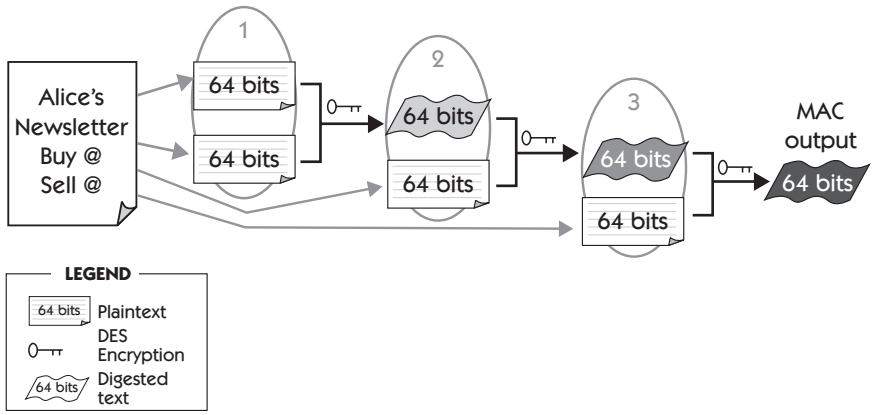

Figure 14-14 Making a MAC from Alice's newsletter. (1) Two 64-bit chunks are combined to make a new 64-bit chunk. DES encryption is applied to the new chunk. (2) The output of step 1 is combined with the third 64-bit chunk. DES encryption is applied to the new chunk. (3) The output of step 2 is combined with the fourth 64-bit chunk. DES encryption is applied to the new chunk. The process repeats until the last 64-bit chunk is combined and DES encryption is applied. The last chunk is the MAC.

8. Although it is not important for our simple overview, if the input file is less than 64 bits or if the file is not evenly divisible by 64, bits are added to the end of the last chunk.

Digest Speed Comparisons

Table 14-1 shows some throughput comparisons of software implementations.

For a speed yardstick, DES is listed first. Hashed MAC (HMAC), the topic of the next section, is a new kind of digest method that combines keyed and non-keyed message digests.

Table 14-1 Comparing the throughput of message digest functions. (HMAC is discussed in the next section.) Source: www.eskimo.com/~weidai.

Method	Megabytes / Sec
DES	7
MD5	57
SHA-1	25
RIPE-160	24
DMAC-RC6	18
HMAC/MD5	56

House of Cards

Here's an interesting rub: Although modern cryptography is dependent on one-way functions (OWFs), there's no proof that one-way functions actually exist[9]—in other words, that there aren't trapdoors. It's possible that all of modern cryptography may be a house of cards! But don't relive your Y2K fears; there are OWFs that are provably as secure as other "hard" math problems. The insecurity comes from the fact that just because mathematicians believe that some problems are hard (and time-consuming) isn't proof that they really are.

Hashed MAC

As shown in Table 14-1, MACs execute much more slowly than non-keyed digest functions such as SHA-1or MD5. So in the mid-1990s, cryptographers proposed a few different ways to combine MACs with non-keyed digest functions to speed up processing.

9. Without making some additional assumptions.

In a 1996 paper, "Keying Hash Functions for Message Authentication," cryptographers Mihir Bellare, Ran Canetti, and Hugo Krawczyk proposed two ways to combine MAC and non-keyed hash functions. One of those methods, HMAC, has become the de facto standard.

HMACs use a secret key and a non-keyed hash function. As of this writing, most HMACs use SHA-1 or MD5. HMACs are as secure as MACs. As substantiation for that claim, current real-world systems such as SSL (Chapter 20) and IPsec (Chapter 21) have standardized on HMAC.

An additional attractive feature of HMAC is that it's designed to easily install and use various non-keyed hash functions. This means that if SHA-1 or MD5 or both are successfully cryptanalyzed, HMAC can be easily retooled to use other, more secure non-keyed hash functions. HMAC has been likened to putting together a home entertainment center; if a particular component fails or becomes obsolete, you simply replace that component.

Review

There are two types of message digests. Keyed, or message authentication codes (MACs), depend on secret keys for security. Non-keyed codes are known by two names: message integrity codes (MICs) and modification detection codes (MDCs). Another name for MIC or MDC is cryptographic checksum.

The security of MICs and MDCs depends on three assurances: one-wayness, weak collision resistance, and strong collision resistance.

The need for processing speed prompted the invention of another message digest function. HMAC, the new kid on the block, combines a MAC (secret key) and an MDC (hash function) to make a secure and more rapid message digest.