*At cats' green on the Sunday he took the message from the inside of the pillar and added Peter Moran's name to the two names already printed there in the "Brontosaur" code. The message now read: "Leviathan to Dragon: Martin Hillman, Trevor Allan, Peter Moran: observe and tail." What was the good of it John hardly knew. He felt better, he felt that at last he had made an attack on Peter Moran instead of waiting passively and effecting no retaliation. Besides, what was the use of being in possession of the key to the codes if he never took advantage of it?*

—***Talking to Strange Men*, Ruth Rendell**

Perhaps the most confusing area of network security is that of message authentication and the related topic of digital signatures. The attacks and countermeasures become so convoluted that practitioners in this area begin to remind one of the astronomers of old, who built epicycles on top of epicycles in an attempt to account for all contingencies. Fortunately, it appears that today's designers of cryptographic protocols, unlike those long-forgotten astronomers, are working from a fundamentally sound model.

It would be impossible, in anything less than book length, to exhaust all the cryptographic functions and protocols that have been proposed or implemented for message authentication and digital signatures. Instead, the purpose of this chapter and the next two is to provide a broad overview of the subject and to develop a systematic means of describing the various approaches.

This chapter begins with an introduction to the requirements for authentication and digital signature and the types of attacks to be countered. Then the basic approaches are surveyed, including the increasingly important area of secure hash functions. Specific hash functions are examined in Chapter 12.

## 11.1 AUTHENTICATION REQUIREMENTS

In the context of communications across a network, the following attacks can be identified:

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent

acknowledgments of message receipt or nonreceipt by someone other than the message recipient.

4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
7. **Source repudiation:** Denial of transmission of message by source.
8. **Destination repudiation:** Denial of receipt of message by destination.

Measures to deal with the first two attacks are in the realm of message confidentiality and are dealt with in Part One. Measures to deal with items 3 through 6 in the foregoing list are generally regarded as message authentication. Mechanisms for dealing specifically with item 7 come under the heading of digital signatures. Generally, a digital signature technique will also counter some or all the attacks listed under items 3 through 6. Dealing with item 8 may require a combination of the use of digital signatures and a protocol designed to counter this attack.

In summary, message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

## 11.2 AUTHENTICATION FUNCTIONS

Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

This section is concerned with the types of functions that may be used to produce an authenticator. These may be grouped into three classes, as follows:

- **Message encryption:** The ciphertext of the entire message serves as its authenticator
- **Message authentication code (MAC):** A public function of the message and a secret key that produces a fixed-length value that serves as the authenticator
- **Hash function:** A public function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

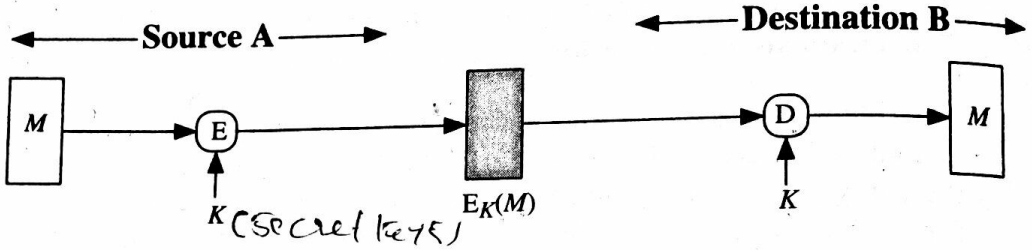
We now briefly examine each of these topics; MACs and hash functions are then examined in greater detail in Sections 11.3 and 11.4.

## Message Encryption

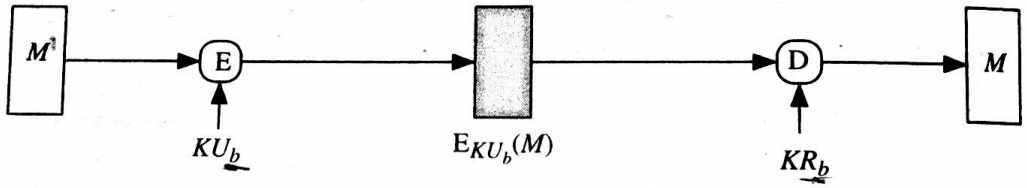
Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

### Symmetric Encryption

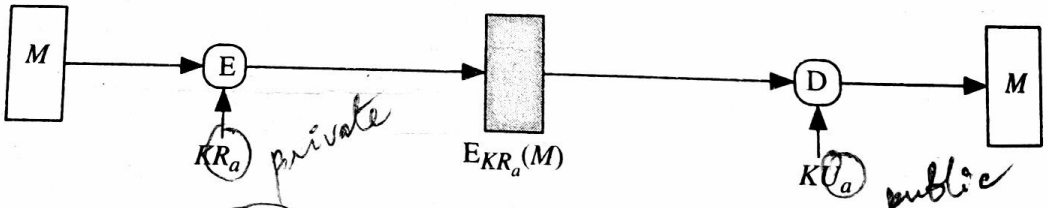
Consider the straightforward use of symmetric encryption (Figure 11.1a). A message $M$ transmitted from source A to destination B is encrypted using a secret key $K$ shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.



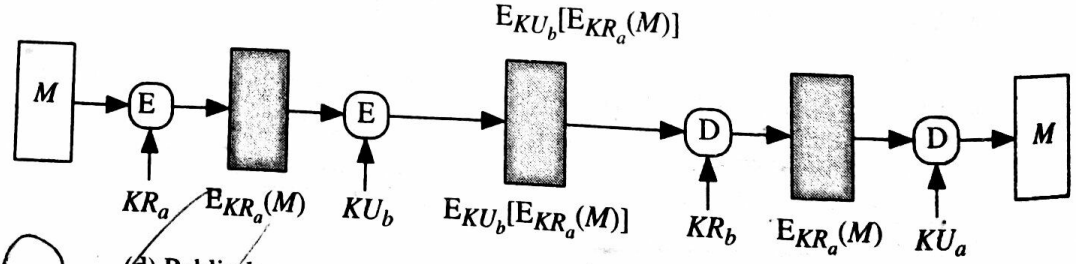

(a) Symmetric encryption: confidentiality and authentication




(b) Public-key encryption: confidentiality




(c) Public-key encryption: authentication and signature




(d) Public-key encryption: confidentiality, authentication, and signature

**Figure 11.1** Basic Uses of Message Encryption

In addition, we may say that B is assured that the message came was generated by A. Why? The message must have come from A because A is the only other party that possesses $K$ and therefore the only other party with the information necessary to construct ciphertext that can be decrypted with $K$. Furthermore, if $M$ is recovered, B knows that none of the bits of $M$ have been altered, because an opponent that does not know $K$ would not know how to alter bits in the ciphertext to produce desired changes in the plaintext.

So we may say that symmetric encryption provides authentication as well as confidentiality. However, this flat statement needs to be qualified. Consider exactly what is happening at B. Given a decryption function D and a secret key $K$, the destination will accept *any* input $X$ and produce output $Y = D_K(X)$. If $X$ is the ciphertext of a legitimate message $M$ produced by the corresponding encryption function, then $Y$ is some plaintext message $M$. Otherwise, $Y$ will likely be a meaningless sequence of bits. There may need to be some automated means of determining at B whether $Y$ is legitimate plaintext and therefore must have come from A.

The implications of the line of reasoning in the preceding paragraph are profound from the point of view of authentication. Suppose the message $M$ can be any arbitrary bit pattern. In that case, there is no way to determine automatically, at the destination, whether an incoming message is the ciphertext of a legitimate message. This conclusion is incontrovertible: If $M$ can be any bit pattern, then regardless of the value of $X$, $Y = D_K(X)$ is *some* bit pattern and therefore must be accepted as authentic plaintext.

Thus, in general, we require that only a small subset of all possible bit patterns is considered legitimate plaintext. In that case, any spurious ciphertext is unlikely to produce legitimate plaintext. For example, suppose that only one bit pattern in $10^6$ is legitimate plaintext. Then the probability that any randomly chosen bit pattern, treated as ciphertext, will produce a legitimate plaintext message is only $10^{-6}$.

For a number of applications and encryption schemes, the desired conditions prevail as a matter of course. For example, suppose that we are transmitting English-language messages using a Caesar cipher with a shift of one ($K = 1$). A sends the following legitimate ciphertext:

```
nbsftfbupbutboeepftfbupbutboemjuumfmbnctfbujwz
```

B decrypts to produce the following plaintext:

```
mareseatoatsanddoeseatoatsandlittlelambseativy
```

A simple frequency analysis confirms that this message has the profile of ordinary English. On the other hand, if an opponent generates the following random sequence of letters,

```
zuvrsoevgqxlzwigamdvnmhpmccxiuureosfbcebtqxsxq
```

this decrypts to

```
ytuqrndufpwkyvhfzlcumlgolbbwhttqdnreabdaspwrwp
```

which does not fit the profile of ordinary English.

It may be difficult to determine *automatically* if incoming ciphertext decrypts to intelligible plaintext. If the plaintext is, say, a binary object file or digitized X-rays, determination of properly formed and therefore authentic plaintext may been

difficult. Thus, an opponent could achieve a certain level of disruption simply by issuing messages with random content purporting to come from a legitimate user.

One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function. We could, for example, append an error-detecting code, also known as a frame check sequence (FCS) or checksum, to each message before encryption, as illustrated in Figure 11.2a. A prepares a plaintext message $M$ and then provides this as input to a function F that produces an FCS. The FCS is appended to $M$ and the entire block is then encrypted. At the destination, B decrypts the incoming block and treats the results as a message with an appended FCS. B applies the same function F to attempt to reproduce the FCS. If the calculated FCS is equal to the incoming FCS, then the message is considered authentic. It is unlikely that any random sequence of bits would exhibit the desired relationship.

Note that the order in which the FCS and encryption functions are performed is critical. The sequence illustrated in Figure 11.2a is referred to in [DIFF79] as internal error control, which the authors contrast with external error control (Figure 11.2b). With internal error control, authentication is provided because an opponent would have difficulty generating ciphertext that, when decrypted, would have valid error control bits. If instead the FCS is the outer code, an opponent can construct messages with valid error-control codes. Although the opponent cannot know what the decrypted plaintext will be, he or she can still hope to create confusion and disrupt operations.

An error-control code is just one example; in fact, any sort of structuring added to the transmitted message serves to strengthen the authentication capabil-

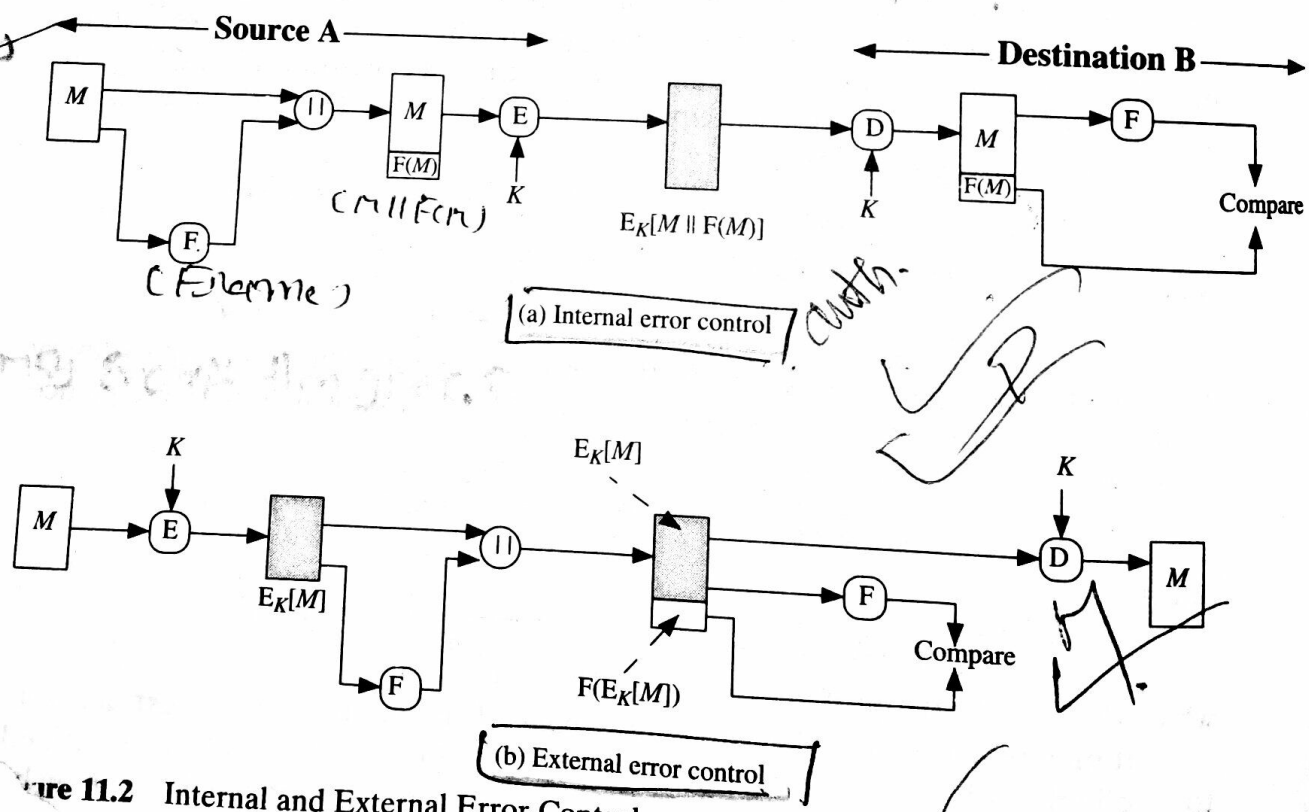



**Figure 11.2** Internal and External Error Control

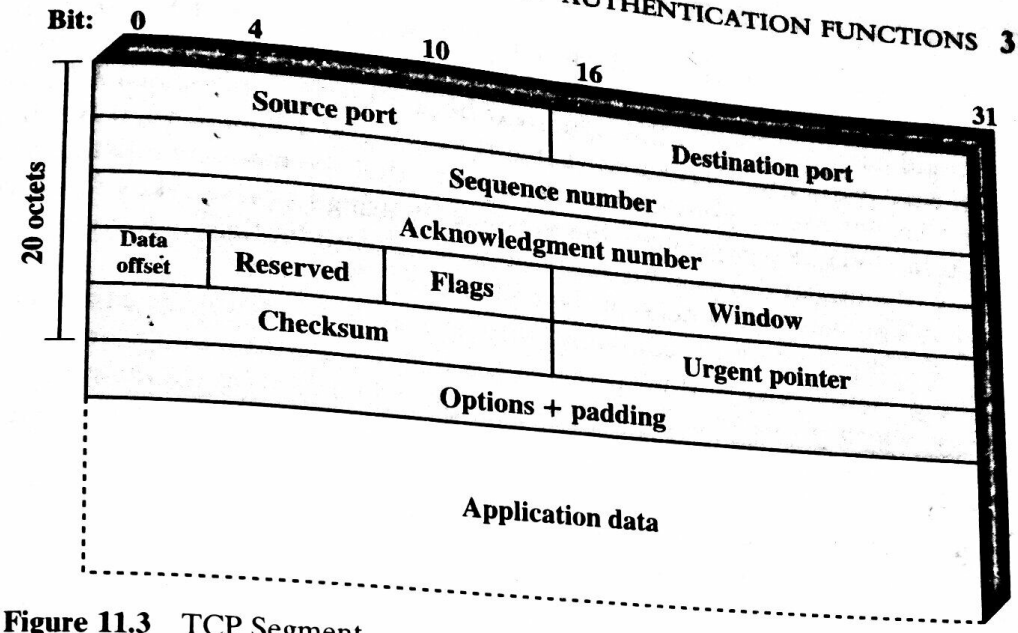



**Figure 11.3** TCP Segment

ity. Such structure is provided by the use of a communications architecture consisting of layered protocols. As an example, consider the structure of messages transmitted using the TCP/IP protocol architecture. Figure 11.3 shows the format of a TCP segment, illustrating the TCP header. Now suppose that each pair of hosts shared a unique secret key, so that all exchanges between a pair of hosts used the same key, regardless of application. Then one could simply encrypt all of the datagram except the IP header (see Figure 7.5). Again, if an opponent substituted some arbitrary bit pattern for the encrypted TCP segment, the resulting plaintext would not include a meaningful header. In this case, the header includes not only a checksum (which covers the header) but other useful information, such as the sequence number. Because successive TCP segments on a given connection are numbered sequentially, encryption assures that an opponent does not delay, misorder, or delete any segments.

### Public-Key Encryption

The straightforward use of public-key encryption (Figure 11.1b) provides confidentiality but not authentication. The source (A) uses the public key $KU_b$ of the destination (B) to encrypt $M$. Because only B has the corresponding private key $KR_b$, only B can decrypt the message. This scheme provides no authentication because any opponent could also use B's public key to encrypt a message, claiming to be A.

To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt (Figure 11.1c). This provides authentication using the same type of reasoning as in the symmetric encryption case: The message must have come from A because A is the only party that possesses $KR_a$ and therefore the only party with the information necessary to construct ciphertext that can be decrypted with $KU_a$. Again, the same reasoning as before applies: There must be some internal structure to the plaintext so that the receiver can distinguish between well-formed plaintext and random bits.

Assuming there is such structure, then the scheme of Figure 11.1c does provide authentication. It also provides what is known as digital signature.[1] Only A could have constructed the ciphertext because only A possesses $KR_a$. Not even B, the recipient, could have constructed the ciphertext. Therefore, if B is in possession of the ciphertext, B has the means to prove that the message must have come from A. In effect, A has "signed" the message by using its private key to encrypt.

Note that this scheme does not provide confidentiality. Anyone in possession of A's public key can decrypt the ciphertext.

To provide both confidentiality and authentication, A can encrypt M first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality (Figure 11.1d). The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

Table 11.1 summarizes the confidentiality and authentication implications of these various approaches to message encryption.

## Message Authentication Code

An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key $K$. When A has a message to send to B, it calculates the MAC as a function of the message and the key: $\text{MAC} = C_K(M)$, where

$$\text{MAC} = C_K(M)$$

$M$ = input message
C = MAC function
$K$ = shared secret key
MAC = message authentication code

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC (Figure 11.4a). If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

[1] This is not the way in which digital signatures are constructed, as we shall see, but the principle is the same.

**Table 11.1** Confidentiality and Authentication Implications of Message Encryption (see Figure 11.1)

| |
|---|
| $A \rightarrow B: E_K[M]$<br>• Provides confidentiality<br>—Only A and B share K<br>• Provides a degree of authentication<br>—Could come only from A<br>—Has not been altered in transit<br>—Requires some formatting/redundancy<br>• Does not provide signature<br>—Receiver could forge message<br>—Sender could deny message<br>(a) Symmetric encryption |
| $A \rightarrow B: E_{KU_b}[M]$<br>• Provides confidentiality<br>—Only B has $KR_b$ to decrypt<br>• Provides no authentication<br>—Any party could use $KU_b$ to encrypt message and claim to be A<br>(b) Public-Key (asymmetric) Encryption |
| $A \rightarrow B: E_{KR_a}[M]$<br>• Provides authentication and signature<br>—Only A has $KR_a$ to encrypt<br>—Has not been altered in transit<br>—Requires some formatting/redundancy<br>—Any party can use $KU_a$ to verify signature<br>(c) Public-key encryption: authentication and signature |
| $A \rightarrow B: E_{KU_b}[E_{KR_a}(M)]$<br>• Provides confidentiality because of $KU_b$<br>• Provides authentication and signature because of $KR_a$<br>(d) Public-key encryption: confidentiality, authenticaton, and signature |

3. If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many-to-one function. The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys. If an $n$-bit MAC is used, then there are $2^n$ possible MACs, whereas there are $N$ possible messages with $N >> 2^n$. Furthermore, with a $k$-bit key, there are $2^k$ possible keys.

For example, suppose that we are using 100-bit messages and a 10-bit MAC. Then there are a total of $2^{100}$ different messages but only $2^{10}$ different MACs. So, on average, each MAC value is generated by a total of $2^{100}/2^{10} = 2^{90}$ different messages.

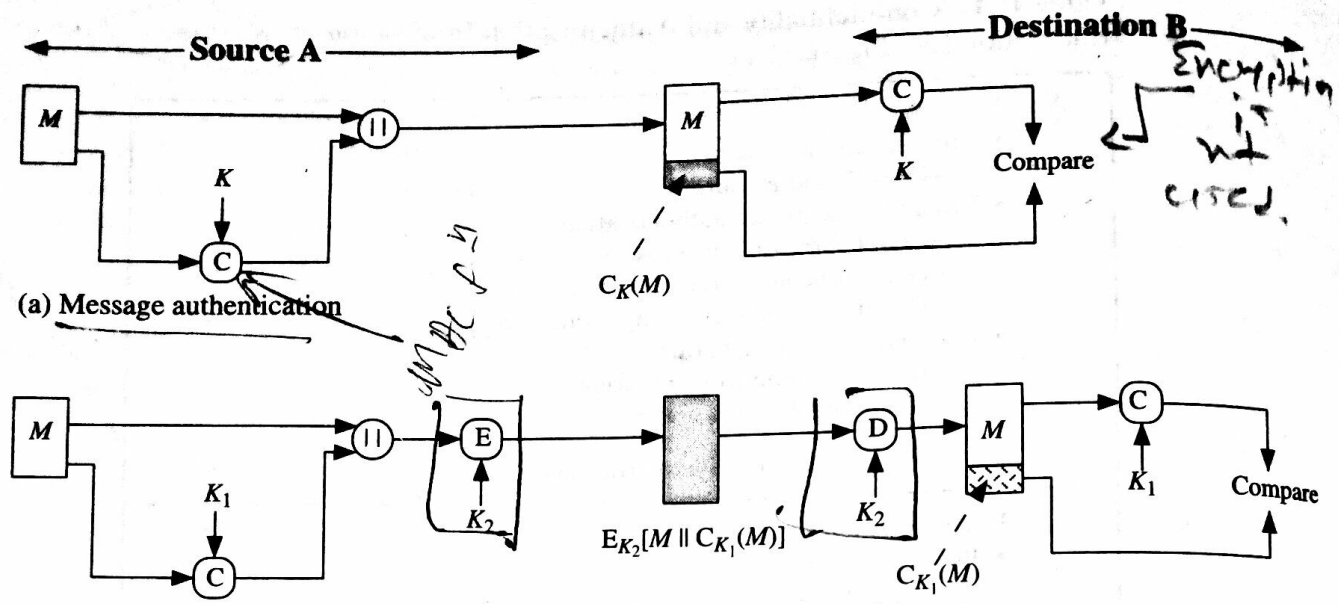



**Figure 11.4** Basic Uses of Message Authentication Code (MAC)

If a 5-bit key is used, then there are $2^5 = 32$ different mappings from the set of messages to the set of MAC values.

It turns out that because of the mathematical properties of the authentication function, it is less vulnerable to being broken than encryption.

The process depicted in Figure 11.4a provides authentication but not confidentiality, because the message as a whole is transmitted in the clear. Confidentiality can be provided by performing message encryption either after (Figure 11.4b) or before (Figure 11.4c) the MAC algorithm. In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver. In the first case, the MAC is calculated with the message as input and is then concatenated to the message. The entire block is then encrypted. In the second case, the message is encrypted first. Then the MAC is calculated using the resulting ciphertext and is concatenated to the ciphertext to form the transmitted block. Typically, it is preferable to tie the authentication directly to the plaintext, so the method of Figure 11.4b is used.

Because symmetric encryption will provide authentication and because it is widely used with readily available products, why not simply use this instead of a separate message authentication code? [DAVI89] suggests three situations in which a message authentication code is used:

**Table 11.2** Basic Uses of Message Authentication Code C (see Figure 11.4)

| |
|---|
| $A \rightarrow B: M \| C_K(M)$<br>• Provides authentication<br>—Only A and B share $K$<br><br>(a) Message authentication |
| $A \rightarrow B: E_{K_2}[M \| C_{K_1}(M)]$<br>• Provides authentication<br>—Only A and B share $K_1$<br>• Provides confidentiality<br>—Only A and B share $K_2$<br><br>(b) Message authentication and confidentiality: authentication tied to plaintext |
| $A \rightarrow B: E_{K_2}[M] \| C_{K_1}(E_{K_2}[M])$<br>• Provides authentication<br>—Using $K_1$<br>• Provides confidentiality<br>—Using $K_2$<br><br>(c) Message authentication and confidentiality: authentication tied to ciphertext |

MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value. The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.

Figure 11.5 illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows:

**a.** The message plus concatenated hash code is encrypted using symmetric encryption. This is identical in structure to the internal error control strategy shown in Figure 11.2a. The same line of reasoning applies: Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

**b.** Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality. Note that the combination of hashing and encryption results in an overall function that is, in fact, a MAC (Figure 11.4a). That is, $E_K[H(M)]$ is a function of a variable-length message $M$ and a secret key $K$, and it produces a fixed-size output that is secure against an opponent who does not know the secret key.

**c.** Only the hash code is encrypted, using public-key encryption and using the sender's private key. As with (b), this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.

**d.** If confidentiality as well as a digital signature is desired, then the message plus the public-key-encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.

**e.** This technique uses a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S. A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recompute the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

**f.** Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

When confidentiality is not required, methods (b) and (c) have an advantage over those that encrypt the entire message in that less computation is required. Nevertheless, there has been growing interest in techniques that avoid encryption (Figure 11.5e). Several reasons for this interest are pointed out in [TSUD92]:

- Encryption software is quite slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.

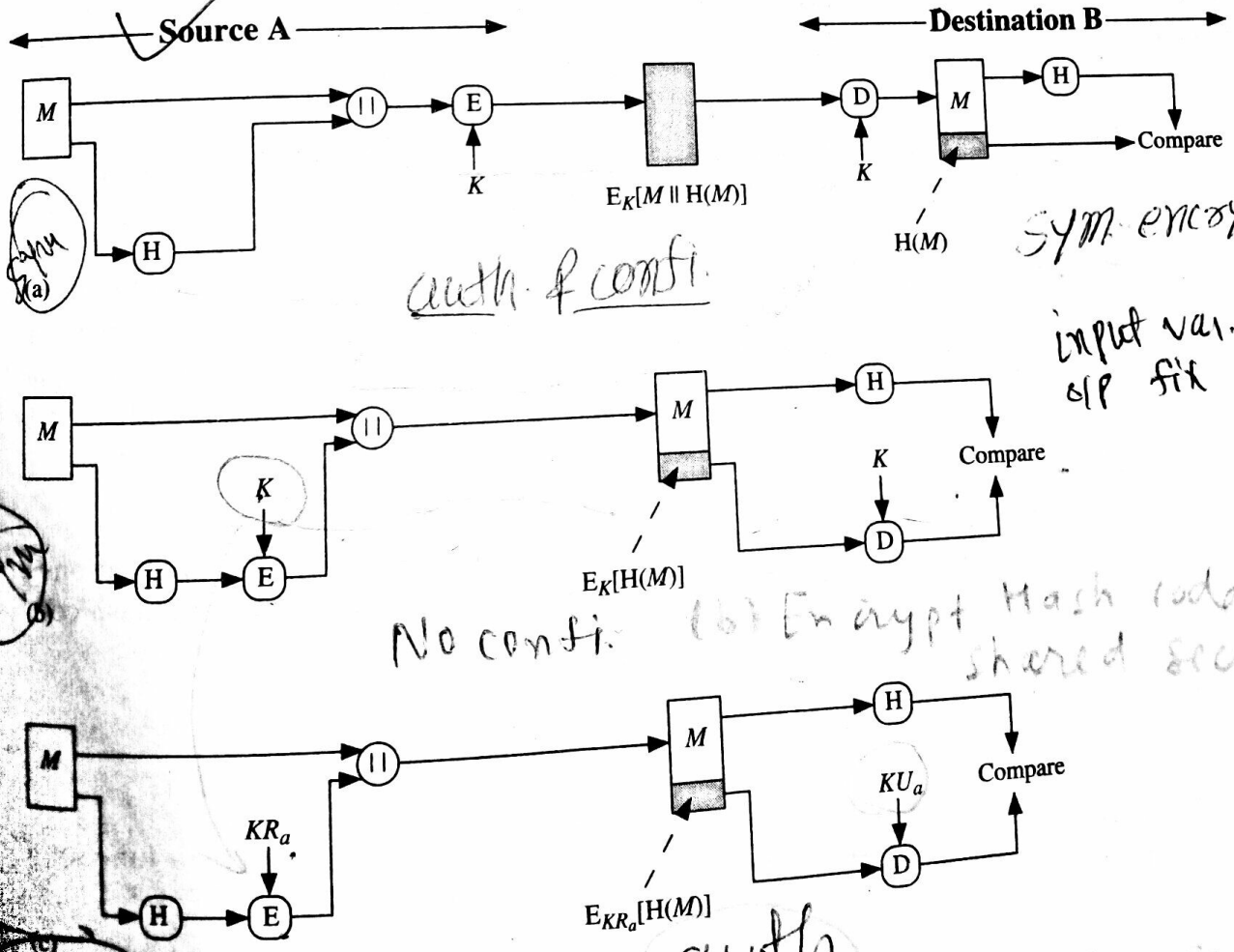



**Figure 11.5** Basic Uses of Hash Function

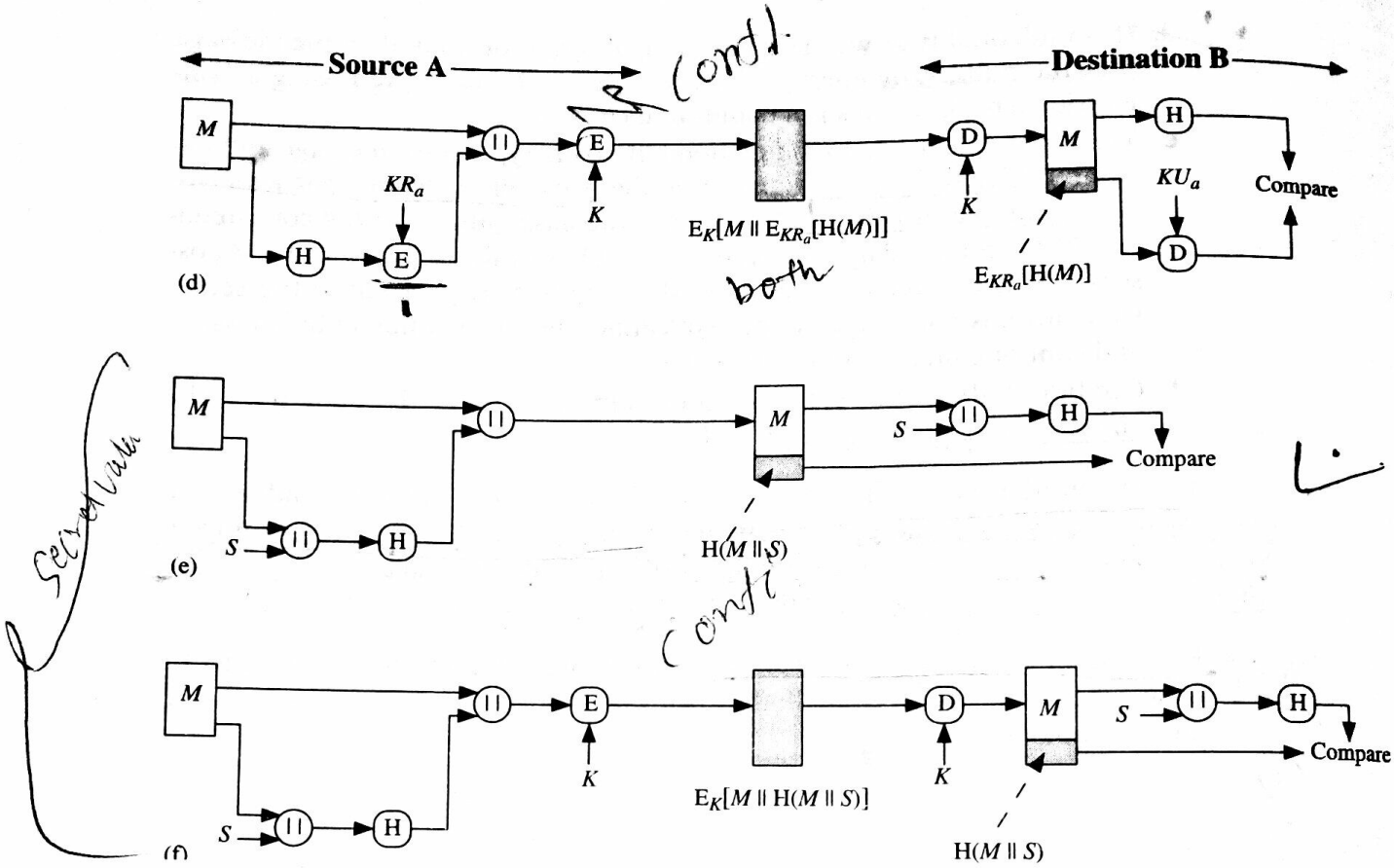



**Figure 11.5** Basic Uses of Hash Function *(Continued)*

- Encryption hardware costs are not negligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
- Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
- Encryption algorithms may be covered by patents. Some encryption algorithms, such as the RSA public-key algorithm, are patented and must be licensed, adding a cost.
- Encryption algorithms are subject to U.S. export control.

Table 11.3 summarizes the confidentiality and authentication implications of the approaches illustrated in Figure 11.5. We next examine MACs and hash codes in more detail.

## 11.3 MESSAGE AUTHENTICATION CODES

A MAC, also known as a cryptographic checksum, is generated by a function C of the form

$$MAC = C_K(M)$$

**Table 11.3** Basic Uses of Hash Function H (see Figure 11.5)

| | |
|---|---|
| A → B: $E_K[M \| H(M)]$<br>• Provides confidentiality<br>—Only A and B share $K$<br>• Provides authentication<br>—H(M) is cryptographically protected<br>(a) Encrypt message plus hash code | A → B: $E_K[M \| E_{KR_a}[H(M)]]$<br>• Provides authentication and digital signature<br>• Provides confidentiality<br>—Only A and B share $K$<br>(d) Encrypt result of (c)—shared secret key |
| A → B: $M \| E_K[H(M)]$<br>• Provides authentication<br>—$H(M)$ is cryptographically protected<br>(b) Encrypt hash code—shared secret key | A → B: $M \| H(M \| S)$<br>• Provides authentication<br>—Only A and B share $S$<br>(e) Compute hash code of message plus secret value |
| A → B: $M \| E_{KR_a}[H(M)]$<br>• Provides authentication and digital signature<br>—$H(M)$ is cryptographically protected<br>—Only A could create $E_{KR_a}[H(M)]$<br>(c) Encrypt hash code—sender's private key | A → B: $E_K[M \| H(M) \| S]$<br>• Provides authentication<br>—Only A and B share $S$<br>• Provides confidentiality<br>—Only A and B share $K$<br>(f) Encrypt result of (e) |

where $M$ is a variable-length message, $K$ is a secret key shared only by sender and receiver, and $C_K(M)$ is the fixed-length authenticator. The MAC is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the MAC.

In this section, we review the requirements for the function C and then examine a specific example. Another example is discussed in Chapter 12.

## Requirements for MACs

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require $2^{(k-1)}$ attempts for a $k$-bit key. In particular, for a ciphertext-only attack, the opponent, given ciphertext $C$, would perform $P_i = D_{K_i}(C)$ for all possible key values $K_i$ until a $P_i$ was produced that matched the form of acceptable plaintext.

In the case of a MAC, the considerations are entirely different. In general, the MAC function is a many-to-one function, due to the many-to-one nature of the function. Using brute-force methods, how would an opponent attempt to discover a key? If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose $k > n$; that is, suppose that the key size is greater than the MAC size. Then, given a known $M_1$ and $MAC_1$, with $MAC_1 = C_{K_1}(M_1)$, the cryptanalyst can perform $MAC_i = C_{K_i}(M_1)$ for all possible key values $K_i$. At least one key is guaranteed to produce a match of $MAC_i = MAC_1$. Note that a total of $2^k$ MACs will be produced, but there are only $2^n < 2^k$ different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has

no way of knowing which is the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:

- **Round 1**
  Given: $M_1$, $\text{MAC}_1 = C_K(M_1)$
  Compute $\text{MAC}_i = C_{K_i}(M_1)$ for all $2^k$ keys
  Number of matches $\approx 2^{(k-n)}$

- **Round 2**
  Given: $M_2$, $\text{MAC}_2 = C_K(M_2)$
  Compute $\text{MAC}_i = C_{K_i}(M_2)$ for the remaining $2^{(k-n)}$ keys
  Number of matches $\approx 2^{(k-2\times n)}$

and so on. On average, $\alpha$ rounds will be needed if $k = \alpha \times n$. For example, if an 80-bit key is used and the MAC is 32 bits long, then the first round will produce about $2^{48}$ possible keys. The second round will narrow the possible keys to about $2^{16}$ possibilities. The third round should produce only a single key, which must be the one used by the sender.

If the key length is less than or equal to the MAC length, then it is likely that a first round will produce a single match. It is possible that more than one key will produce such a match, in which case the opponent would need to perform the same test on a new (message, MAC) pair.

Thus, a brute-force attempt to discover the authentication key is no less effort and may be more effort than that required to discover a decryption key of the same length. However, other attacks that do not require the discovery of the key are possible.

Consider the following MAC algorithm. Let $M = (X_1 \| X_2 \| \ldots \| X_m)$ be a message that is treated as a concatenation of 64-bit blocks $X_i$. Then define

$$\Delta(M) = X_1 \oplus X_2 \oplus \ldots \oplus X_m$$
$$C_K(M) = E_K[\Delta(M)]$$

where $\oplus$ is the exclusive-OR (XOR) operation and the encryption algorithm is DES in electronic codebook mode. Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes $\{M \| C_K(M)\}$, a brute-force attempt to determine $K$ will require at least $2^{56}$ encryptions. But the opponent can attack the system by replacing $X_1$ through $X_{m-1}$ with any desired values $Y_1$ through $Y_{m-1}$ and replacing $X_m$ with $Y_m$, where $Y_m$ is calculated as follows:

$$Y_m = Y_1 \oplus Y_2 \oplus \cdots \oplus Y_{m-1} \oplus \Delta(M)$$

The opponent can now concatenate the new message, which consists of $Y_1$ through $Y_m$, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length $64 \times (m - 1)$ bits can be fraudulently inserted.

Thus, in assessing the security of a MAC function, we need to consider the types of attacks that may be mounted against it. With that in mind, let us state the require-

ments for the function. Assume that an opponent knows the MAC function C but does not know $K$. Then the MAC function should have the following properties:

1. If an opponent observes $M$ and $C_K(M)$, it should be computationally infeasible for the opponent to construct a message $M'$ such that $C_K(M') = C_K(M)$.
2. $C_K(M)$ should be uniformly distributed in the sense that for randomly chosen messages, $M$ and $M'$, the probability that $C_K(M) = C_K(M')$ is $2^{-n}$, where $n$ is the number of bits in the MAC.
3. Let $M'$ be equal to some known transformation on $M$. That is, $M' = f(M)$. For example, f may involve inverting one or more specific bits. In that case, $\Pr[C_K(M) = C_K(M')] = 2^{-n}$.

The first requirement speaks to the earlier example, in which an opponent is able to construct a new message to match a given MAC, even though the opponent does not know and does not learn the key. Requirement (2) deals with the need to thwart a brute-force attack based on chosen plaintext. That is, if we assume that the opponent does not know $K$ but does have access to the MAC function and can present messages for MAC generation, then the opponent could try various messages until finding one that matches a given MAC. If the MAC function exhibits uniform distribution, then a brute-force method would require, on average, $2^{(n-1)}$ attempts before finding a message that fits a given MAC.

The final requirement dictates that the authentication algorithm should not be weaker with respect to certain parts or bits of the message than others. If this were not the case, then an opponent who had $M$ and $C_K(M)$ could attempt variations on $M$ at the known "weak spots" with a likelihood of early success at producing a new message that matched the old MAC.

## Message Authentication Code Based on DES

One of the most widely used MACs, referred to as the Data Authentication Algorithm, is based on DES. The algorithm is both a FIPS publication (FIPS PUB 113) and an ANSI standard (X9.17).

The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES (Figure 3.12) with an initialization vector of zero. The data (e.g., message, record, file, or program) to be authenticated are grouped into contiguous 64-bit blocks: $D_1, D_2, \ldots, D_N$. If necessary, the final block is padded on the right with zeroes to form a full 64-bit block. Using the DES encryption algorithm, E, and a secret key, $K$, a data authentication code (DAC) is calculated as follows (Figure 11.6):

$$
\begin{aligned}
O_1 &= E_K(D_1) \\
O_2 &= E_K(D_2 \oplus O_1) \\
O_3 &= E_K(D_3 \oplus O_2) \\
&\vdots \\
O_N &= E_K(D_N \oplus O_{N-1})
\end{aligned}
$$

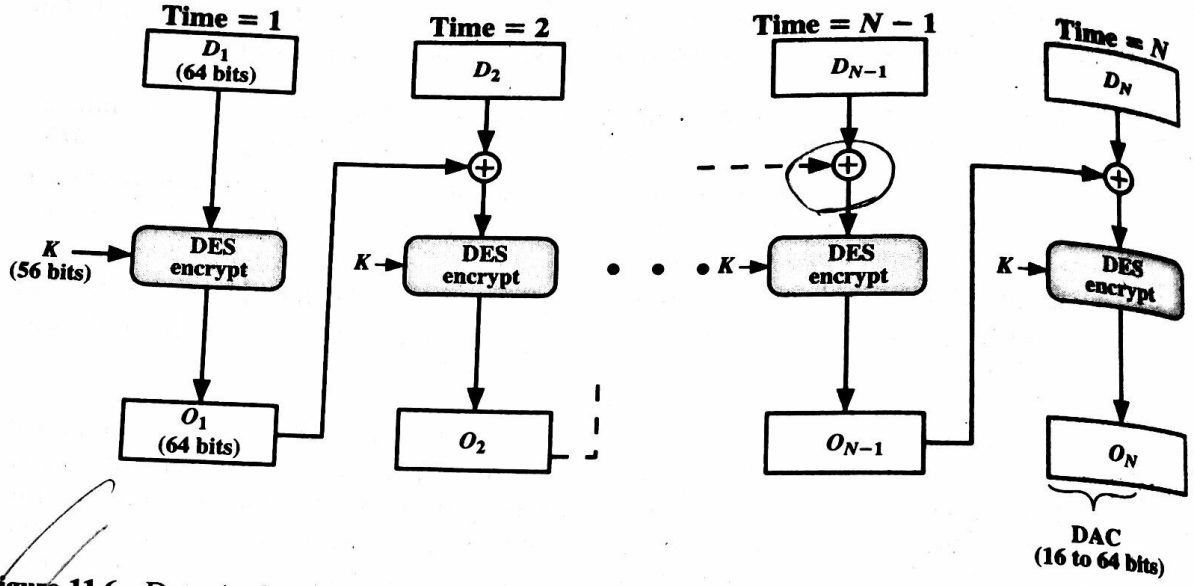



**Figure 11.6** Data Authentication Algorithm (FIPS PUB 113)

The DAC consists of either the entire block $O_N$ or the leftmost $M$ bits of the block, with $16 \leq M \leq 64$.

This algorithm appears to meet the requirements specified earlier.

## 11.4 HASH FUNCTIONS

A hash value $h$ is generated by a function H of the form

$$h = \mathrm{H}(M)$$

where $M$ is a variable-length message and $\mathrm{H}(M)$ is the fixed-length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the hash value. Because the hash function itself is not considered to be secret, some means is required to protect the hash value (Figure 11.5).

We begin by examining the requirements for a hash function to be used for message authentication. Because hash functions are, typically, quite complex, it is useful to examine next some very simple hash functions to get a feel for the issues involved. We then look at several approaches to hash function design.

### Requirements for a Hash Function

The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties (adapted from a list in [NECH92]):

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. $H(x)$ is relatively easy to compute for any given $x$, making both hardware and software implementations practical.
4. For any given value $h$, it is computationally infeasible to find $x$ such that $H(x) = h$. This is sometimes referred to in the literature as the **one-way** property.
5. For any given block $x$, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$. This is sometimes referred to as **weak collision resistance**.
6. It is computationally infeasible to find any pair $(x, y)$ such that $H(x) = H(y)$. This is sometimes referred to as **strong collision resistance**.[2]

The first three properties are requirements for the practical application of a hash function to message authentication.

The fourth property is the one-way property: It is easy to generate a code given a message but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value (Figure 11.5e). The secret value itself is not sent; however, if the hash function is not one way, an attacker can easily discover the secret value: If the attacker can observe or intercept a transmission, the attacker obtains the message $M$ and the hash code $C = H(S_{AB} \| M)$. The attacker then inverts the hash function to obtain $S_{AB} \| M = H^{-1}(C)$. Because the attacker now has both $M$ and $S_{AB} \| M$, it is a trivial matter to recover $S_{AB}$.

The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used (Figures 11.5b and c). For these cases, the opponent can read the message and therefore generate its hash code. However, because the opponent does not have the secret key, the opponent should not be able to alter the message without detection. If this property were not true, an attacker would be capable of the following sequence: First, observe or intercept a message plus its encrypted hash code; second, generate an unencrypted hash code from the message; third, generate an alternate message with the same hash code.

The sixth property refers to how resistant the hash function is to a class of attack known as the birthday attack, which we examine shortly.

## Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of $n$-bit blocks. The input is processed one block at a time in an iterative fashion to produce an $n$-bit hash function.

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im}$$

[2]Unfortunately, these terms are not used consistently. Alternate terms used in the literature include *one-way hash function* (properties 4 and 5); *collision-resistant hash function* (properties 4, 5, and 6); *weak one-way hash function* (properties 4 and 5); *strong one-way hash function* (properties 4, 5, and 6). The reader must take care in reading the literature to determine the meaning of the particular terms used.

where

$C_i$ = $i$th bit of the hash code, $1 \leq i \leq n$
$m$ = number of $n$-bit blocks in the input
$b_{ij}$ = $i$th bit in $j$th block
$\oplus$ = XOR opseration

Figure 11.7 illustrates this operation; it produces a simple parity for each bit position and is known as a longitudinal redundancy check. It is reasonably effective for random data as a data integrity check. Each $n$-bit hash value is equally likely. Thus, the probability that a data error will result in an unchanged hash value is $2^{-n}$. With more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. So if a 128-bit hash value is used, instead of an effectiveness of $2^{-128}$, the hash function on this type of data has an effectiveness of $2^{-112}$.

A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed. The procedure can be summarized as follows:

1. Initially set the $n$-bit hash value to zero.
2. Process each successive $n$-bit block of data as follows:
   a. Rotate the current hash value to the left by one bit.
   b. XOR the block into the hash value.

This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input. Figure 11.8 illustrates these two types of hash functions for 16-bit hash values.

Although the second procedure provides a good measure of data integrity, it is virtually useless for data security when an encrypted hash code is used with a plaintext message, as in Figures 11.5b and c. Given a message, it is an easy matter to produce a new message that yields that hash code: Simply prepare the desired alternate message and then append an $n$-bit block that forces the new message plus block to yield the desired hash code.

Although a simple XOR or rotated XOR (RXOR) is insufficient if only the hash code is encrypted, you may still feel that such a simple function could be use-

| | Bit 1 | Bit 2 | . . . | Bit $n$ |
|---|---|---|---|---|
| **Block 1** | $b_{11}$ | $b_{21}$ | | $b_{n1}$ |
| **Block 2** | $b_{12}$ | $b_{22}$ | | $b_{n2}$ |
| | ⋮ | ⋮ | ⋮ | ⋮ |
| **Block m** | $b_{1m}$ | $b_{2m}$ | | $b_{nm}$ |
| **Hash code** | $C_1$ | $C_2$ | | $C_n$ |

**Figure 11.7** Simple Hash Function Using Bitwise XOR

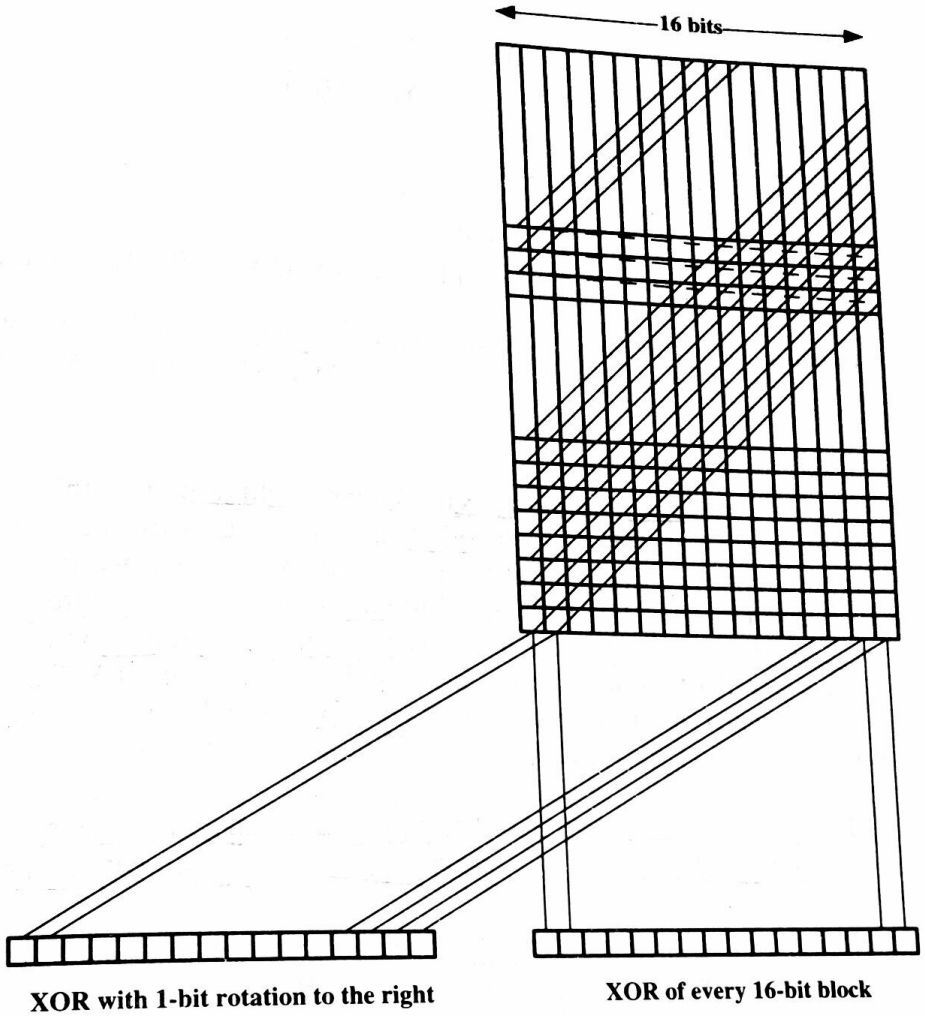



**Figure 11.8** Two Simple Hash Functions

ful when the message as well as the hash code are encrypted (Figure 11.5a). But you must be careful. A technique originally proposed by the National Bureau of Standards used the simple XOR applied to 64-bit blocks of the message and then an encryption of the entire message that used the cipher block chaining (CBC) mode. We can define the scheme as follows: Given a message consisting of a sequence of 64-bit blocks $X_1, X_2, \ldots, X_N$, define the hash code $C$ as the block-by-block XOR of all blocks and append the hash code as the final block:

$$C = X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

Next, encrypt the entire message plus hash code, using CBC mode to produce the encrypted message $Y_1, Y_2, \ldots, Y_{N+1}$. [JUEN85] points out several ways in which the ciphertext of this message can be manipulated in such a way that it is not detectable by the hash code. For example, by the definition of CBC (Figure 3.12), we have

$$X_1 = IV \oplus D_K(Y_1)$$
$$X_i = Y_{i-1} \oplus D_K(Y_i)$$
$$X_{N+1} = Y_N \oplus D_K(Y_{N+1})$$

But $X_{N+1}$ is the hash code:

$$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$
$$= (IV \oplus D_K(Y_1)) \oplus (Y_1 \oplus D_K(Y_2)) \oplus \ldots \oplus (Y_{N-1} \oplus D_K(Y_N))$$

Because the terms in the preceding equation can be XORed in any order, it follows that the hash code would not change if the ciphertext blocks were permuted.

### Birthday Attacks

Suppose that a 64-bit hash code is used. One might think that this is quite secure. For example, if an encrypted hash code C is transmitted with the corresponding unencrypted message $M$ (Figure 11.5b or c), then an opponent would need to find an $M'$ such that $H(M') = H(M)$ to substitute another message and fool the receiver. On average, the opponent would have to try about $2^{63}$ messages to find one that matches the hash code of the intercepted message [see Appendix 11A, Equation (11.1)].

However, a different sort of attack is possible, based on the birthday paradox (Appendix 11A). Yuval proposed the following strategy [YUVA79]:

1. The source, A, is prepared to "sign" a message by appending the appropriate $m$-bit hash code and encrypting that hash code with A's private key (Figure 11.5c).
2. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. The opponent prepares an equal number of messages, all of which are variations on the fraudulent message to be substituted for the real one.
3. The two sets of messages are compared to find a pair of messages that produces the same hash code. The probability of success, by the birthday paradox, is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.
4. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known.

Thus, if a 64-bit hash code is used, the level of effort required is only on the order of $2^{32}$ [see Appendix 11A, Equation (11.7)].

The generation of many variations that convey the same meaning is not difficult. For example, the opponent could insert a number of "space-space-backspace" character pairs between words throughout the document. Variations could then be

Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --} Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the} Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --} responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches} in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs} to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the {latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum} of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document} as proof of identity. An order with his signature, which is {appended / attached} {authorizes / allows} you to charge the cost to this company at the {above / head office} address. We {fully / --} expect that our {level / volume} of orders will increase in the {following / next} year and {trust / hope} that the new appointment will {be / prove} {advantageous / an advantage} to both our companies.

**Figure 11.9** A Letter in $2^{37}$ Variations [DAVI89]

generated by substituting "space-backspace-space" in selected instances. Alternatively, the opponent could simply reword the message but retain the meaning. Figure 11.9 [DAVI89] provides an example.

The conclusion to be drawn from this is that the length of the hash code should be substantial. We discuss this further in Section 11.5.

## Block Chaining Techniques

A number of proposals have been made for hash functions based on using a cipher block chaining technique, but without the secret key. One of the first such proposals was that of Rabin [RABI78]. Divide a message $M$ into fixed-size blocks $M_1$, $M_2, \ldots, M_N$ and use a symmetric encryption system such as DES to compute the hash code $G$ as follows:

$$H_0 = \text{initial value}$$
$$H_i = \mathrm{E}_{M_i}[H_{i-1}]$$
$$G = H_N$$

This is similar to the CBC technique, but in this case there is no secret key. As with any hash code, this scheme is subject to the birthday attack, and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable.

Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings. Here is the scenario; we assume that the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is $m$ bits long:

1. Use the algorithm defined at the beginning of this subsection to calculate the unencrypted hash code $G$.
2. Construct any desired message in the form $Q_1, Q_2, \ldots, Q_{N-2}$.
3. Compute $H_i = \mathrm{E}_{Q_i}[H_{i-1}]$ for $1 \leq i \leq (N-2)$.
4. Generate $2^{m/2}$ random blocks; for each block $X$, compute $\mathrm{E}_X[H_{N-2}]$. Generate an additional $2^{m/2}$ random blocks; for each block $Y$, compute $\mathrm{D}_Y[G]$, where D is the decryption function corresponding to E.
5. Based on the birthday paradox, with high probability there will be an $X$ and $Y$ such that $\mathrm{E}_X[H_{N-2}] = \mathrm{D}_Y[G]$.
6. Form the message $Q_1, Q_2, \ldots, Q_{N-2}, X, Y$. This message has the hash code $G$ and therefore can be used with the intercepted encrypted signature.

This form of attack is known as a "meet in the middle" attack. A number of researchers have proposed refinements intended to strengthen the basic block chaining approach. For example, Davies and Price [DAVI89] describe the following variation:

$$H_i = \mathrm{E}_{M_i}[H_{i-1}] \oplus H_{i-1}$$

Another variation, proposed in [MEYE88], is as follows:

$$H_i = \mathrm{E}_{H_{i-1}}[M_i] \oplus M_i$$

However, both of these schemes have been shown to be vulnerable to a variety of attacks [MIYA90]. More generally, it can be shown that some form of birthday attack will succeed against any hash scheme involving the use of cipher block chaining without a secret key provided that either the resulting hash code is small enough (e.g., 64 bits or less) or that a larger hash code can be decomposed into independent subcodes [JUEN87].

Thus, attention has been directed at finding other approaches to hashing. Many of these have also been shown to have weaknesses [MITC92]. We examine some strong hash functions in Chapter 12.