

3.a 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:

$$\text{MAC} = \text{MAC}(K, M)$$

where

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

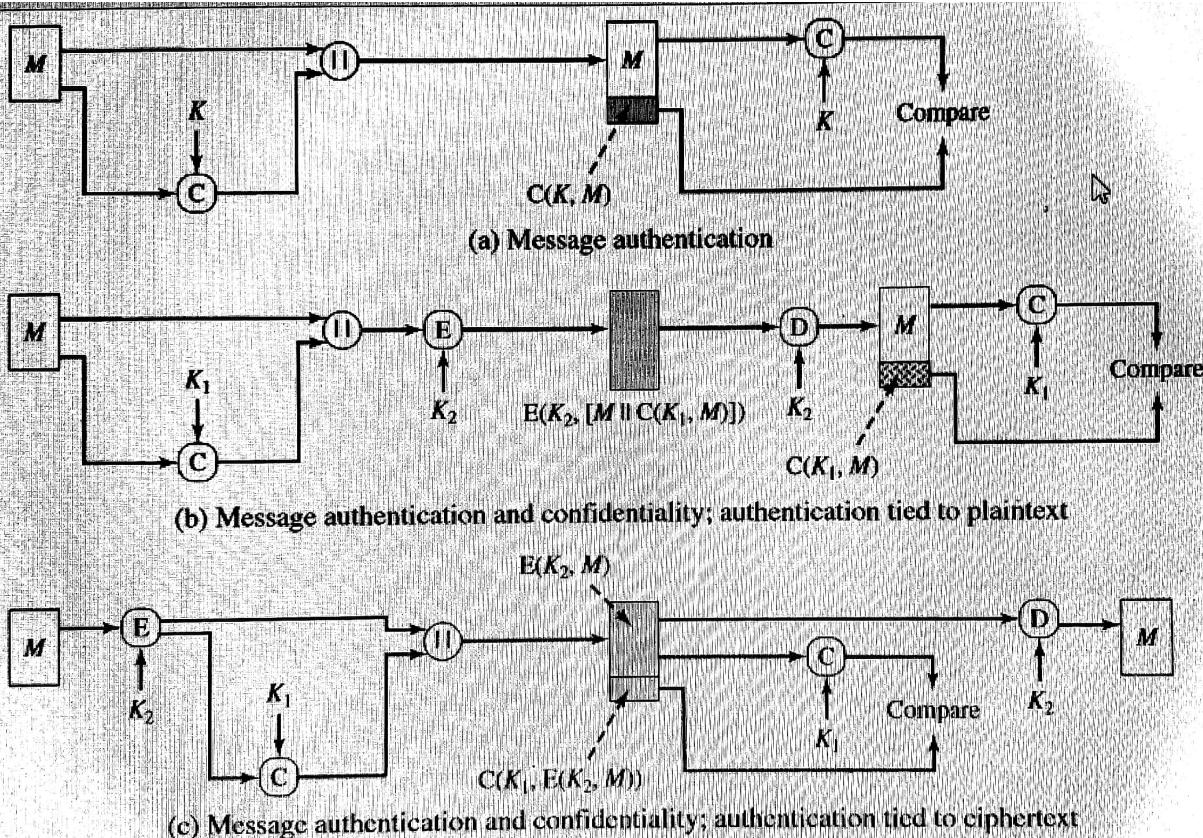


Figure 12.4 Basic Uses of Message Authentication code (MAC)

generate a new MAC. The received MAC is compared to the calculated MAC (Figure 12.4a). 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.
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.

12.3 REQUIREMENTS FOR MESSAGE AUTHENTICATION CODES

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

$$T = \text{MAC}(K, M)$$

where M is a variable-length message, K is a secret key shared only by sender and receiver, and $\text{MAC}(K, M)$ is the fixed-length authenticator, sometimes called a tag. The tag 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 tag.

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 , performs $P_i = D(K_i, C)$ for all possible key values K_i until a P_i is produced that matches 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

If we employ such an approach, 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 T_1 , with $T_1 = \text{MAC}(K, M_1)$, the cryptanalyst can perform $T_i = \text{MAC}(K_i, M_1)$ for all possible key values k_i . At least one key is guaranteed to produce a match of $T_i = T_1$. Note that a total of 2^k tags will be produced, but there are only $2^n < 2^k$ different tag values. Thus, a number of keys will produce the correct tag and the opponent has no

12.3 / REQUIREMENTS FOR MESSAGE AUTHENTICATION CODES 373

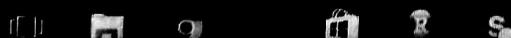
way of knowing which is the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack.

• Round 1

Given: $M_1, T_1 = \text{MAC}(K, M_1)$

Compute $T_i = \text{MAC}(K_i, M_1)$ for all 2^k keys

Number of matches $\approx 2^{(k-n)}$



- Round 2

Given: $M_2, T_2 = \text{MAC}(K, M_2)$

Compute $T_i = \text{MAC}(K_i, M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1

Number of matches $\approx 2^{(k-2n)}$



And so on. On average, α rounds will be needed if $k = \alpha \times n$. For example, if an 80-bit key is used and the tag is 32 bits, 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 tag 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, tag) 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 $M = (X_1 \parallel X_2 \parallel \dots \parallel X_m)$ be a message that is treated as a concatenation of 64-bit blocks X_i . Then define

$$\begin{aligned}\Delta(M) &= X_1 \oplus X_2 \oplus \dots \oplus X_m \\ \text{MAC}(K, M) &= E(K, \Delta(M))\end{aligned}$$

© 2014 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has determined that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

but does not know K . Then the MAC function should satisfy the following requirements.

1. If an opponent observes M and $\text{MAC}(K, M)$, it should be computationally infeasible for the opponent to construct a message M' such that

$$\text{MAC}(K, M') = \text{MAC}(K, M)$$

2. $\text{MAC}(K, M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M' , the probability that $\text{MAC}(K, M) = \text{MAC}(K, M')$ is 2^{-n} , where n is the number of bits in the tag.
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,

$$\Pr [\text{MAC}(K, M) = \text{MAC}(K, M')] = 2^{-n}$$