Source :

Information Security

Mark Stamp

Section 5.5 , pages 93-94

5.5 HMAC

Recall that for message integrity we can compute a message authentication code, or MAC, using a block cipher in cipher block chaining (CBC) mode. The MAC is the final encrypted block, which is known as the CBC residue. Since a hash function yields a different value if the input changes, we should be able to use a hash to verify message integrity. But we can't send the message M along with its hash $h(M)$, since an attacker could simply replace M with M' and $h(M)$ with $h(M')$. However, if we make the hash depend on a symmetric key, then we can compute a *hashed MAC*, or HMAC.

How should we mix the key into the HMAC? Two obvious approaches are $h(K, M)$ and $h(M, K)$. Suppose we choose to compute the HMAC as $h(K, M)$. There is a potential problem with this approach. Most cryptographic hashes hash the message in blocks. For MD5, SHA–1, and Tiger, the block size used is 512 bits. As a result, if $M = (B_1, B_2)$, where each B_i is 512 bits, then

$$h(M) = F(F(A, B_1), B_2) = F(h(B_1), B_2) \tag{5.2}$$

for some function F, where A is a fixed initial constant. For example, in the Tiger hash, the function F consists of the outer round illustrated in Figure 5.1, with each B_i corresponding to a 512-bit block of the input and A corresponding to the 192-bit initial values of (a, b, c).

If $M' = (M, X)$, Trudy might be able to use *equation 5.2* to find $h(K, M')$ from $h(K, M)$ without knowing K, since, for K, M, and X of the appropriate size,

$$h(K, M') = h(K, M, X) = F(h(K, M), X) \tag{5.3}$$

where the function F is known.

Is $h(M, K)$ better? It does prevent the previous attack. However, if it should happen that there is a collision, that is, if there exists some M' with $h(M') = h(M)$, then by *equation 5.2*, we have

$$h(M, K) = F(h(M), K) = F(h(M'), K) = h(M', K) \tag{5.4}$$

provided that M and M' are each a multiple of the block size. This is certainly not as serious of a concern as the previous case, since if such a collision occurs, the hash function is insecure. But if we can eliminate this attack, then we should do so.

In fact, we can prevent both of these potential problems by slightly modifying the method used to mix the key into the hash. As described in RFC 2104 [133], the approved method for computing an HMAC is as follows. Let B be the block length of hash, in bytes. For MD5, SHA–1, and Tiger, $B = 64$. Next, define

$$\text{ipad} = \texttt{0x36} \text{ repeated } B \text{ times}$$

and

$$\text{opad} = \texttt{0x5C} \text{ repeated } B \text{ times.}$$

Then the HMAC of M is

$$\text{HMAC}(M, K) = H(K \oplus \text{opad}, H(K \oplus \text{ipad}, M))$$

which thoroughly mixes the key into the resulting hash. An HMAC can be used in place of a MAC for message integrity. HMACs also have several other uses, some of which we'll see in later chapters