CBC-MAC

The basic CBC-MAC construction is as follows:

$Gen(1^n)$: upon input 1^n, choose a uniformly distributed string $k \leftarrow \{0, 1\}^n$

$Mac_k(m)$: upon input key $k \in \{0, 1\}n$ and a message m of length $l \cdot n$, do the following:

1. Denote $m = m_1, \ldots, m_l$ where each mi is of length n, and set $t0 = 0^n$.

2. For i = 1 to l, set $t_i \leftarrow Fk\ (t_{i-1} \oplus m_i)$ where $F : \{0, 1\}* \times \{0, 1\}* \rightarrow \{0, 1\}*$ is a function.

3. Output t^l

$Vrfy_k$ (m, t): upon input key $k \in \{0, 1\}^n$, a message m of length $l \cdot n$ and a tag t of length n, output 1 if and only if $t = Mac_k(m)$

Let l be any fixed value. If F is a pseudorandom function such that for every $k \in \{0, 1\}^n$ the function F_k maps n-bit strings to n-bit strings, then CBC-MAC Construction is a fixed-length MAC with length parameter $l \cdot n$ that is existentially unforgeable under a chosen-message attack.

In order to obtain a secure MAC via the CBC construction for variable-length messages, Construction 4.7 must be modified. This can be done in a number of ways. Three possible options that have been proven secure are:

1. Apply the pseudorandom function (block cipher) to the block length l of the input message in order to obtain a key k_l. Then, compute the basic CBC-MAC using the key k_l and send the resulting tag along with the block length.

2. Prepend the message with its block length l, and then compute the CBC-MAC (the first block contains the number of blocks to follow). We stress that appending the message with its block length is not secure.

3. Choose two different keys $k_1 \leftarrow \{0, 1\}^n$ and $k_2 \leftarrow \{0, 1\}^n$. Then, compute the basic CBC-MAC using k_1 ; let t_1 be the result. The output MAC-tag is defined to be $t = F_{k2}\ (t_1)$.

We note that the third option has the advantage that it is not necessary to know the message length before starting to compute the MAC. Its disadvantage is that it requires

two keys. However, at the expense of two additional applications of the pseudorandom function, it is possible to store a single key k and then derive keys $k_1 = F_k(1)$ and $k_2 = F_k(2)$ at the beginning of the computation.