1 Construction for the CMA-Secure MAC

We have a PRF that works with l-bit long messages

$$F_K : \{0,1\}^l \rightarrow \{0,1\}^L$$

(1)

It can authenticate l-bit long messages.

$$m \in \{0,1\}^l$$

(2)

$$\nu = F_K(m)$$

(3)

Output $$(m, \mu)$$

(4)

In practice, messages are arbitrarily long. We cannot compute separate MACs for each l-bit long block of the message separately using a PRF. This is mainly because such a construction will not be secure; a MAC on a message $$(m_1, m_2, \ldots, m_n)$$ (where each $$m_i$$ is l-bit long) is also a MAC on any permutation (with respect to different blocks) of the message.

1.1 CBC-MAC

One can encode longer messages by dividing a message into l-bit long blocks and chaining them, similar to CBC-DES.

To verify a message and MAC pair, the receiver computes the MAC on the message and compare it with the received MAC value.

Differences between CBC MAC scheme and CBC encryption are as follows:

- CBC MAC produces one output block for any length message, whereas CBC encryption produces $$N$$ l-bit output blocks for $$N$$ l-bit plaintext blocks.

- Initialisation is not required for CBC MAC (i.e, IVs are not used), since the MAC is deterministic and no randomization is needed as in CBC encryption.

**Theorem 1** If $$F_K$$ is a PRF (or CMA-secure), then CBC-MAC is also CMA-secure.
Proof. We will prove the contrapositive $\neg\text{CBC-MAC} \rightarrow \neg\text{PRF}(F_K)$

We will prove if CBC-MAC is not CMA-secure, then $F_K$ is not a PRF. In other words, we will prove that if there exists an adversary $A$ which can break the CMA-security of CBC-MAC, then we can construct $B$ which can break the PRF property of $F_K$.

$A$ sends a signing query to $B$. As $B$ cannot send the entire message to the challenger, it splits the message into $l$-bit blocks and sends one block at a time. $B$ then XORs the returned response from the challenger with the next message block. The process is repeated till all blocks are processed in this manner. The challenger’s response to the last block is the MAC for query message $m_i$. $B$ then sends the MAC to $A$. Thus, $B$ simulates the action of signing oracle to $A$. $A$ continues to send $q$ such messages to $B$.

Eventually, $A$ decides it is ready to create a forgery. $A$ then sends message $M$, consisting of $n$ blocks $M_1, M_2, \ldots, M_n$, and MAC $\mu$ to $B$. $B$ forwards the message blocks in the same manner as was done was previously queried messages $m_i$, and finally obtains the corresponding MAC. If the computed MAC equals $\mu$, then $B$ sends $A$, and the challenger, $d = 1$. Otherwise, it sends $A$, and the Challenger, $d = 0$.

\[
\text{Adv}^\text{PRF}_F(B) = Pr(E_{\text{exp}}^{\text{PRF}}(B) = 1) - Pr(E_{\text{exp}}^{\text{PRF}^{-1}}(B) = 1)
\]
\[
= \text{Adv}^\text{CMA}_{\text{CBC-MAC}}(A) - \Delta
\]
\[
\Delta = Pr(\text{atleast one collision in } q \text{ queries})
\]

Adversary $A$ recorded $q$ responses $(y_1, y_2, \ldots, y_q)$, when it queried the signing oracle $B$ on messages $m_1, m_2, \ldots, m_q$. Note that we are concerned with the case when the challenger was in World 0, i.e., when the PRF challenger was set to be a random function denoted by $RF$. If any of the two responses collide, i.e., if $y_{i\alpha} = y_{j\alpha}$ (for some $i, j \leq q$), then
For the above equality to be true, the messages must be equal, since $RF$ is a random function, and thus

$$y_{i,n} - 1 \oplus m_{i,n} = y_{j,n} - 1 \oplus m_{j,n}$$

Then, for any binary string $B$ of length $l$-bits, we have

$$y_{i,n-1} \oplus m_{i,n} = y_{j,n-1} \oplus m_{j,n}$$

The adversary can make another query $m_{i,1}, m_{i,2}, ..., m_{i,n} \oplus B$, and receive in return MAC $\sigma_i'$
And then the adversary can create a forgery by sending \( m_{j,1}, m_{j,2}, \ldots, m_{j,n} \oplus B \) as the message and \( \sigma'_i \) as the corresponding MAC.

Therefore, if the responses of any two previously queried \( q \) messages collide, then the adversary can produce a valid forgery, just by issuing an additional query. This means that the probability that the adversary wins even when it is fed the responses of a random function is \( \Delta \leq \frac{q(q-1)}{2^{2l}} \) (from the upper bound of the birthday attack).

Completing the proof

\[
\begin{align*}
Adv_F^{PRF}(B) &= Adv_{CBC-MAC}^{CMA}(A) - \Delta \\
Adv_F^{PRF}(B) &= Adv_{CBC-MAC}^{CMA}(A) - \frac{q(q-1)}{2 \cdot 2^l} \\
\Rightarrow Adv_{CBC-MAC}^{CMA}(A) &= Adv_F^{PRF}(B) + \frac{q(q-1)}{2 \cdot 2^l}
\end{align*}
\]

When \( \Delta \leq negl \) and \( Adv_F^{PRF} \) is negl, \( Adv_{CBC-MAC}^{CMA}(A) \leq negl \)

For \( \Delta \leq negl \), one can set \( q=2^{40} \) and \( l=2^{160} \), for example.

\[ \text{Proposition 1} \quad CBC-MAC \text{ can be used to authenticate only a fixed length messages (i.e it can not be used to authenticate arbitrarily long messages of variable lengths)} \]

\[ \text{Proof.} \quad \text{If the oracle accepted arbitrarily long messages, the adversary } A \text{ can create a forgery by performing the concatenation } (m_{11}, m_{12}, y_{12}) \text{ as shown in the Figure below (extension attack)} \]
Solution to Fixed-Length Restriction on CBC MAC

To prevent forgery for arbitrarily long messages, we will incorporate the number of message blocks (or the length of the message) into the block computation. The key used for PRF computation can be set to $F_K(l)$ instead of $K$, where $l$ is the number of message blocks. This variant remains CMA secure and it allows to authenticate variable length messages.

From the figure 1.1, we see that the new authenticator is $\mu = F_{F_{l}(l)}(m)$ where $l$ is the number of message blocks with equal size.
If the adversary attempts to create a forgery $(m_{11}, m_{12}, y_{12})$, the oracle will not accept because the MAC is now computed as $y_{12} = F_{F_{K}(2)}(F_{F_{K}(2)}(m_{11}) \oplus m_{12})$. 

9-5
2 Nested MAC (NMAC)

An NMAC function is a MAC construction based on hash functions which used two key components $K_1$ and $K_2$. First the message is compressed using a cascading hash function, $H_{K_2}^*(\cdot)$, then the output of the cascading hash function is fed to a compression function $h_{K_1}^*(\cdot)$.

\[
NMAC_{K_1, K_2}(m) = h_{K_1}^*(H_{K_2}^*(m))
\]  

(5)

$H_{K_1}^*(\cdot)$: cascaded hash function where $K_1$ replaces IV of the underlying hash function $h_{K_2}^*(\cdot)$: compression function where $K_2$ replaces IV of the underlying compression function.

MAC constructions such as $H(m, K)$, $H(K, m)$ and $H(K_1, m, K_2)$ are not used because they are either not secure or are not provably secure.
Theorem 2 If $h^*$ is CMA-secure MAC and $H^*$ is collision resistant, then NMAC is CMA-secure.

Proof. If there exists an adversary $A$ who can break the CMA security of NMAC, then we can construct $B$ who can break the CMA security of $h^*$. The reduction is given in Figure ??.

\[
\text{Adv}_{h^*}^{\text{CMA}}(B) = \Pr((H_{K_2}^*(M), \mu) \text{ is a valid forgery on } h^*) \\
= 1 - \Pr((H_{K_2}^*(M), \mu) \text{ is not a valid forgery on } h^*) \\
= 1 - [\Pr((M, \mu) \text{ is not a valid forgery on } \text{NMAC}) \\
+ \Pr(H_{K_2}^*(M) \text{ collides onto one of the previously queried messages})] \\
= \text{Adv}_{\text{NMAC}}^{\text{CMA}}(A) - \Pr(H^* \text{ is not collision resistant}) \\
\text{Adv}_{\text{NMAC}}^{\text{CMA}}(A) = \text{Adv}_{h^*}^{\text{CMA}}(B) + \text{Adv}_{H^*}^{\text{CR}}(B)
\]
If we set $\text{Adv}^{\text{CMA}}_{h}(B) \leq \text{negl}$ and $\text{Adv}^{\text{CR}}_{h}(B) \leq \text{negl}$, i.e. $\frac{1}{2^{m}}$, then $\text{Adv}^{\text{CMA}}_{\text{NMAC}}(A) = \frac{2}{2^{m}}$. Therefore, $\text{Adv}^{\text{CMA}}_{\text{NMAC}}(A) = \text{negl}$.

**Note:** NMAC is not used in practice because it is difficult to replace the IV to the key components when the source code for the underlying hash function is not accessible.
3 HMAC

HMAC is an efficient MAC construction based on existing hash function (no need to modify the underlying IV value)

\[ HMAC_K(m) = NMAC_{K_1,K_2}(m) \]  

(6)

where \( K_1 = h(K \oplus opad) \) and \( K_2 = h(K \oplus ipad) \).

Note that HMAC is quite efficient as it only requires 3 more compression calls than H(x). This makes HMAC a very fast construction.

**Theorem 3** HMAC is CMA secure if NMAC is CMA secure.

The proof follows from the security of NMAC under the assumption that the keys \( K_1 \) and \( K_2 \) are random. In practice, these keys are made random enough as follows. The default values of opad and ipad are: \( opad = 0x5c5c5c5c...5c5c \) \( ipad = 0x363636...3636 \) These values are included in HMAC mainly to increase the hamming distance between the keying data in the inner and outer calls to H(). This ensures that the two key values are as distinct as possible while using the same key in both locations.