# 0 CBC-MAC

Write the basic construction of CBC-MAC

**Answer:** Let $F$ be a pseudorandom function, and fix a length function $\ell > 0$. The basic CBC-MAC construction is as follows:

- **Mac**: on input key $k \in \{0,1\}^n$ and a message $m$ of length $\ell(n) \cdot n$, do the following:

  1. Parse $m$ as $m = m_1, m_2, \ldots, m_\ell$ where each $m_i$ is of length $n$.
  2. Set $t_0 := 0^n$. Then for $i = 1$ to $ell$: Set $t_i := F_k(t_{i-1} \oplus m_i)$

  Output $t_\ell$ as the tag

- **Vrfy**: on input key $k \in \{0,1\}^n$, a message $m$, and a tag $t$, do: If $m$ is not of length $\ell(n) \cdot n$ then output 0. Otherwise, output 1 iff $t = Mac_k(m)$

# 1 Merkle-Damgård

Let $h : \{0,1\}^{n+t} \to \{0,1\}^n$ be a fixed-length compression function. Suppose we forgot a few features of Merkle-Damgard and construct $H$ as follows:

- Value $x$ is input.
- Split $x$ into $y_0, x_1, \ldots, x_k$. Where $y_0$ is $n$ bits and $x_i$ (for $i$ = 1,...,$k$) is $t$ bits.. The last piece $x_k$ may be padded with zeroes.
- For $i$ = 1,...,$k$, set $y_i = h(y_{i-1}||x_i)$.
- Output $y_k$.

It's similar to Merkle-Damgård except no IV and the final padding block is missing.

1. Describe an easy way to find two messages that are broken up into the same number of pieces, which have the same hash value under $H$.

**Attack**

(a) Query Oracle to get the key $s$ for the hash function $H$

(b) $m_1 := m_1||t||11$

(c) $m_2 := m_1||t||110$

(d) Output $m_1, m_2$

**Analysis**

- The hash would be computed for $m_1 := m||t||11||0^{t-2}$
- The hash would be computed for $m_2 := m||t||110||0^{t-3}$

So $m_1 \neq m_2$ but $H(m_1) = H(m_2)$ Hence a collision was found.

2. Describe an easy way to find two messages that are broken up into a different number of pieces, which have the same hash value under $H$. *Hint: Pick any string of length n + 2t, and find a shorter string that collides with it.*

**Attack**

(a) Query Oracle to get the key $s$ for the hash function $H$

(b) $m_1 := y_0||x_1||x_2$

(c) Compute $a := H(y_0||x_1)$

(d) $m_2 := a||x_2$

(e) Output $m_1, m_2$

**Analysis**

- The hash would be computed for $H(m_1) := H(y_0||x_1||x_2)$
- The hash would be computed for $H(m_2) := H(a||x_2) = H(y_0||x_1||x_2)$

So $m_1 \neq m_2$ but $H(m_1) = H(m_2)$ Hence a collision was found.

# 2 Hash Functions

I designed $H : \{0,1\} \rightarrow \{0,1\}$. I make $H(x) = x$ if $x$ is $n$-bit string - but assume $H's$ behavior is more complicated on strings of other lengths. This way we know there are no collisions among $n$-bit strings. Is this a good design decision?

No, this is not a good design.

**Attack**

1. Query Oracle to get the key $s$ for the hash function $H$
2. $m_1 :=\in \{0,1\}^{n+x}$
3. Compute $y* := H(m_1)$
4. $m_2 := y*$
5. Output $m_1, m_2$

**Analysis**

- The hash would be computed for $H(m_1) := y*$
- The hash would be computed for $H(y*) := y*$

So $m_1 \neq m_2$ but $H(m_1) = H(m_2)$ Hence a collision was found.
Which is a collision.

# 3 Message Authentication Code

Prove that the following modifications of basic CBC-MAC do not yield a secure MAC (even for fixed-length messages).

1. Mac outputs all blocks $t_1, \ldots, t_\ell$ rather than just $t_\ell$. Verification only checks if $t_\ell$ is correct.

**Attack**

(a) Query Oracle with $m := m_1||m_2$

(b) Get back $t_1,\ t_2$,

(c) Finally output $m* := t_1 \oplus m_2||t_2 \oplus m_1$ and $t* := t_2||t_1$

**Analysis**

(a) When we run Vrfy$(m*, t*)$, we have the following

- We will have Mac$(m*)$ :=Mac$(t_1 \oplus m_2||t_2 \oplus m_1)$
- Which will give $t_2||F_k(t_2 \oplus m_1 \oplus t_2) = t_2||t_1$

The $\mathcal{A}$ will output 1 with probability 1

2. A random initial block is used each time a message is authenticated. That is, choose a uniform $t_0 \in \{0,1\}^n$, run basic CBC-MAC over the "message" $t_0, m_1, \ldots, m_\ell$ and output tag $\langle t_0, t_l \rangle$. Verification is done in a natural way.

**Attack**

(a) Query Oracle with $m' = m_1 || m_2$

(b) Get back $t_0 || t_2$

(c) Pick a arbitrary $t* \xrightarrow{\$} \{0,1\}^n$

(d) Finally Output

- $m* = m_1 \oplus t* \oplus t_0 || m_2$
- $t* := t* || t_2$

**Analysis**

(a) When we run Vrfy$(m*, t*)$, we have the following

- We will have Vrfy$(m*)$ :=Vrfy$(m_1 \oplus t* \oplus t_0 || m_2)$
- Which will give $F_k(m_1 \oplus t_0) || F_k(m_2 \oplus F_K(m_1 \oplus t_0) = t* || t_2$

The $\mathcal{A}$ will output 1 with probability 1

# 4 Digital Signature

Let $(G, S, V)$ be a secure signature scheme with message space $\{0,1\}^n$, and security parameter $\lambda$. Let $(pk_0, sk_0) \leftarrow_{\$} G(1^\lambda)$ and $(pk_1, sk_1) \leftarrow_{\$} G(1^\lambda)$ be two pairs of signing/verification keys. Which of the following are secure signature schemes? Show an attack or prove security.

1. $(S_1, V_1)$

   - Sign. $S_1((sk_0, sk_1), m)$ : Output $(S(sk_0, m), S(sk_1, m))$.
   - Verify. $V_1((pk_0, pk_1), m, (\sigma_0, \sigma_1))$: Output 1 if $(V(pk_0, m, \sigma_0) \vee V(pk_1, m, \sigma_1))$, 0 otherwise.

   The scheme $\tilde{\Pi} := (S_1, V_1)$ is secure.

   **Theorem:** If $\Pi := (G,\ S,\ V)$ is secure then $\tilde{\Pi}$ is secure.

   **Proof by contradiction:** If $\tilde{\Pi}$ is not secure then $\exists$ a PPT Adversary $\mathcal{A}$ that breaks $\Pi$ with a non-negligible probability.

2. $(S_2, V_2)$

   - Sign. $S_2((sk_0, sk_1), (m_L, m_R))$ : Output $(S(sk_0, m_L), S(sk_1, m_R))$.
   - Verify. $V_2((pk_0, pk_1), (m_L, m_R), (\sigma_0, \sigma_1))$: Output 1 if $(V(pk_0, m_L, \sigma_0) \wedge V(pk_1, m_R, \sigma_1))$, 0 otherwise.

**Attack**

1. Query with $m' := m_L||m_L$
   - Get back $\sigma := \sigma_0^0||\sigma_1^0$
2. Query with $m'' := m_R||m_R$
   - Get back $\sigma := \sigma_0^1||\sigma_1^1$
3. Finally output
   - $m* := m_L||m_R$ and
   - $t* := \sigma_0^0||\sigma_1^1$
   - where $m_L \neq m_R$

**Analysis**

1. When we run the Vrfy$(m*, \sigma*)$
   - Vrfy$(m*)$=Vrfy$(m_L||m_R)$=$\sigma_0^0||\sigma_1^1$
   - where $m_L \neq m_R$

The $\mathcal{A}$ will output 1 with probability 1