

TD 6: Message Authentication Codes (corrected version)

Exercise 1.*Insecure MACs*

Let $F : \{0,1\}^t \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a secure pseudo-random function (PRF). Show that each one of the following message authentication codes (MAC) is insecure:

1. To authenticate $m = m_1 \parallel \dots \parallel m_d$ where $m_i \in \{0,1\}^n$ for all i , compute $t = F(k, m_1) \oplus \dots \oplus F(k, m_d)$.

 **Definition :** Secure MAC – existential unforgeability under a chosen message attack. Attacker has access to a signing oracle: $m_i \rightarrow t_i = \text{Sign}(k, m_i)$ for $i \leq q = \text{queries nbr}$. Attacker must produce a new pair $(m, t) \notin (m_i, t_i)_{i=1}^q$, such that $\text{Verify}(k, m, t) = 1$.

Ask a tag for $(m_1 \parallel 0 \parallel \dots \parallel 0)$, $t_1 = F(k, m_1) \oplus F(k, 0)$. Return $m = (0 \parallel \dots \parallel 0 \parallel m_1)$ and t_1 .

2. To authenticate $m = m_1 \parallel \dots \parallel m_d$ with $d < 2^{n/2}$ and $m_i \in \{0,1\}^{n/2}$ for all i , compute

$$t = F(k, \underline{1} \parallel m_1) \oplus \dots \oplus F(k, \underline{d} \parallel m_d),$$

where $\underline{i}$ is an $n/2$ -bit long representation of i , for all $i \leq d$.



Ask tag of $(0 \parallel 0 \parallel \dots \parallel 0)$: $t_0 = \bigoplus_{i \geq 1} F(k, \underline{i} \parallel 0)$.

Ask tag of $(m_1 \parallel 0 \parallel \dots \parallel 0)$: $t_1 = F(k, \underline{1} \parallel m_1) \oplus \bigoplus_{i \geq 2} F(k, \underline{i} \parallel 0)$.

Ask tag of $(0 \parallel m_2 \parallel 0 \parallel \dots \parallel 0)$: $t_2 = F(k, \underline{1} \parallel 0) \oplus F(k, \underline{2} \parallel m_2) \oplus \bigoplus_{i \geq 3} F(k, \underline{i} \parallel 0)$.

Then $(t_0 \oplus t_1 \oplus t_2 = F(k, \underline{1} \parallel m_1) \oplus F(k, \underline{2} \parallel m_2) \oplus \bigoplus_{i \geq 3} F(k, \underline{i} \parallel 0))$ is a valid tag for $(m_1 \parallel m_2 \parallel 0 \parallel \dots \parallel 0)$.

Exercise 2.*CCA Insecurity*

Let us consider the following symmetric encryption scheme, where $F : \{0,1\}^s \times \{0,1\}^n \rightarrow \{0,1\}^\ell$ is a secure PRF. To encrypt a message $m \in \{0,1\}^\ell$ for $\ell \in \mathbb{N}$:

KeyGen(1^λ): Output $k \leftarrow U(\{0,1\}^s)$.

Enc(k, m): Sample $r \leftarrow U(\{0,1\}^n)$ and output $c := (r, F(k, r) \oplus m)$.

Dec($k, c := (c_1, c_2)$): Output $m = c_2 \oplus F(k, c_1)$.

1. Recall the security definition of the CCA-security of an encryption scheme.

 See the lecture.

2. Is this scheme CCA-secure?

 Let $\mathcal{A}$ be the adversary that does the following. It samples two different messages m_0, m_1 and gets an encryption (r^*, c^*) of m_b for a b it has to guess. It then queries the decryption of (r^*, c) for any $(c \neq c^*)$ and gets $F(k, r^*) \oplus c$. With this it can get $F(k, r^*)$ back. And finally it can decrypt c^* and know the value of b . Then this scheme cannot be CCA-secure.

Exercise 3.*CBC-MAC*

Let $F : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ be a PRF, $d > 0$ and $L = nd$. Prove that the following modifications of CBC-MAC (recalled in Figure 1) do not yield a secure fixed-length MAC. Define $t_i := F(K, t_{i-1} \oplus m_i)$ for $i \in [1, d]$ and $t_0 := IV = 0$.

1. Modify CBC-MAC so that a random $IV \leftarrow U(\{0,1\}^n)$ (rather than $IV = 0$) is used each time a tag is computed, and the output is (IV, t_d) instead of t_d alone.

 If an adversary asks for a tag (t_0, t_d) of any $(m_1, \dots, m_d)$, then it can output $(t_0 \oplus x, t_d), (m_1 \oplus x, \dots, m_d)$ as a forgery, as it is a valid pair of a tag and a message. Such an adversary wins everytime and has non-negligible advantage in the unforgeability game.

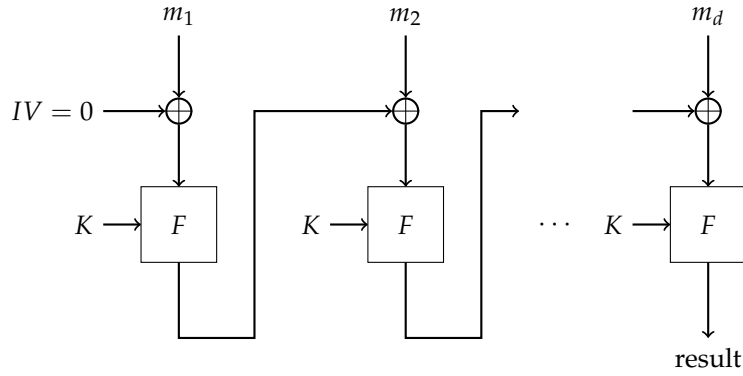


Figure 1: CBC-MAC

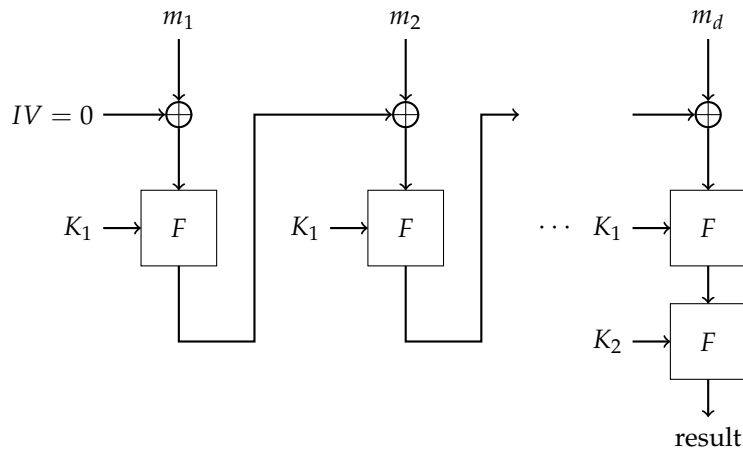


Figure 2: ECBC-MAC

2. Modify CBC-MAC so that all the outputs of F are output, rather than just the last one.



If an adversary asks for a tag $(t_1, t_2, \dots, t_d)$ of any message $(m_1, m_2, \dots, m_d)$, then it can output $(t_2, t_3, \dots, t_d, t_1), (m_2 \oplus t_1, m_3, \dots, m_d, t_d)$ as a forgery as it is a valid pair (tag, message). Such an adversary wins everytime. Indeed, $F(K, m_2 \oplus t_1 \oplus 0) = t_2$ by definition and $F(K, t_d \oplus t_d) = t_1$ since $m_1 = 0$.

We now consider the following ECBC-MAC scheme: let $F : K \times X \rightarrow X$ be a PRF, we define $F_{ECBC} : K^2 \times X^{\leq L} \rightarrow X$ as in Figure 2, where K_1 and K_2 are two independent keys.

If the message length is not a multiple of the block length n , we add a pad to the last block: $m = m_1 \parallel \dots \parallel m_{d-1} \parallel (m_d \parallel \text{pad}(m))$.

3. Show that there exists a padding for which this scheme is not secure.



We could for instance pad with as many 0s as necessary.

Let m of length $< n$. Then, $m \parallel \text{pad}(m) = m \parallel 0 \parallel \text{pad}(m \parallel 0)$. As such we build an adversary for the unforgeability game that:

- asks for a tag for m of length $< n$.
- Gets a tag t .
- Returns the forgery $(m \parallel 0, t)$.

This adversary always wins and as such breaks the unforgeability of the scheme.

For the security of the scheme, the padding must be invertible, and in particular for any message $m_0 \neq m_1$ we need to have $m_0 \parallel \text{pad}(m_0) \neq m_1 \parallel \text{pad}(m_1)$. In practice, the ISO norm is to pad with $10 \dots 0$, and if the message length is a multiple of the block length, to add a new “dummy” block $10 \dots 0$ of length n .

4. Prove that this scheme is not secure if the padding does not add a new “dummy” block if the message length is a multiple of the block length.

 Let $m = m_1 \parallel 100$ of the length of a block, then $m = m_1 \parallel \text{pad}(m_1)$, so any valid tag for m is a valid tag for m_1 .

Remark: The NIST standard is called CMAC, it is a variant of CBC-MAC with three keys (k, k_1, k_2) . If the message length is not a multiple of the block length, then we append the ISO padding to it and then we also XOR this last block with the key k_1 . If the message length is a multiple of the block length, then we XOR this last block with the key k_2 . After that, we perform a last encryption with $F(k, \cdot)$ to obtain the tag.