CS 466: Homework 4

Problem 1. (40 points.) Let D be the set of all strings whose length is a positive multiple of 128. Define hash function $H\colon \{0,1\}^{128} \times D \to \{0,1\}^{128}$ as follows:

Algorithm $H_K(M)$:
  Parse M as $M[1]M[2]\ldots M[m]$
  $C[0] \leftarrow 0^{128}$
  For $i = 1$ to m do:
    $B[i] \leftarrow \mathsf{AES}_K(C[i-1] \oplus M[i])$
    $C[i] \leftarrow \mathsf{AES}_K(B[i] \oplus M[i])$
  Return $C[m]$

As before, above we parse M as consisting of m blocks of 128-bits each. Show that H is not CR by giving a practical adversary A such that its advantage $\mathbf{Adv}^{\mathrm{cr}}_H(A)$ is high, say a constant c. As usual, your adversary should be given in concise pseudocode (30 points) and you should formally analyze its advantage and resource usage (10 points). NB: Your adversary should break the hash function without breaking the underlying blockcipher as a PRF. Such attacks against the underlying blockcipher are not practical and will not receive any points.

Problem 2. (40 points.) Recall that a MAC $\mathcal{T}\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ has a canonical verification algorithm: On inputs m, σ, if $\mathcal{T}(K, m) = \sigma$ then output 1 (accept), else output 0 (reject). In the UF-CMA game, we did not give the adversary access to a verification oracle. It turns out that *black-box* access to a verification oracle does not help the adversary, But suppose the oracle is augmented such that on input m', σ' it not only returns a bit indicating whether $\mathcal{T}(K, m) = \sigma$ but also the *time* it took to compute this equality check. The interpretation is that the adversary gets to know how long it took the receiver to accept or reject. Further, suppose the equality check is implemented in the straightforward way of checking one bit position of $\mathcal{T}(K, m)$ and σ at a time and stopping when a bit position differs.

Part A: (30 points.) Argue that this leads to a *timing attack*: the adversary is able to forge on an arbitrary message of its choice! That is, present pseudocode for an adversary playing this augmented version of the UF-CMA game that outputs a forgery on an arbitrary message m (20 points). Prove that your algorithm is correct (10 points).

Part B: What does this mean for a receiver who implements the canonical verification algorithm in practice?

For clarity, the augmented UF-CMA game referred to above is specified as followed, For a function family $\mathcal{T}\colon \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ define the games below:

proc INITIALIZE
$K \leftarrow\!\!\$\ K$
$S \leftarrow \emptyset$

proc TAG(x)
$x \leftarrow S \cup \{x\}$
Return $\mathcal{T}(K, x)$

proc VER(x, σ)
$acc \leftarrow \mathsf{false}$
If $x \in \mathcal{D}$ and $x \notin S$ and $\mathcal{T}(K, x) = \sigma$ then $acc \leftarrow \mathsf{true}$
Let i be the number of steps it took for the above line to execute
Return (acc, i)

proc FINALIZE(x, σ)
$acc \leftarrow \mathsf{false}$
If $x \in \mathcal{D}$ and $x \notin S$ and $\mathcal{T}(K, x) = \sigma$
  Then $acc \leftarrow \mathsf{true}$
Return acc