Problem 1: One-way functions

Which of the following are one-way functions? For each function that is a one-way function, explain why (no formal proof required). For each function that is not a one-way function, write an attack in Python. (Code for all the functions, including test code is provided in `owf.py`. You only need to fill in the functions `adv` for attacking function $f_i$.)

**Hint:** Out of the four functions, one is a OWF, the other three are not.

**Note:** You may assume that the RSA assumption holds. And that $E_{AES}$ is a PRF.

**Note:** Remember that to break a one-way function, it is sufficient to find some preimage, not necessarily the “true” one that was fed into the one-way function.

(a) $f_1(x) := 0$ for all $x \in \{0,1\}^n$.

(b) $f(N, e, x) := (N, e, x^e \mod N)$ where the domain of $f$ is the set of all $(N, e, x)$ where $N$ is an RSA modulus, $e$ is relatively prime to $N$, and $x \in \{0, \ldots, N-1\}$.

(c) $f(N, e, x) := x^e \mod N$ where the domain of $f$ is the set of all $(N, e, x)$ where $N$ is an RSA modulus, $e$ is relatively prime to $N$, and $x \in \{0, \ldots, N-1\}$.

(d) $f(k, x) := E_{AES}(k, x)$.

Problem 2: Tree-based signatures

This problem refers to the tree-based construction of signature schemes from one-time signatures from Construction 7 in the lecture notes. You may assume that Lamport’s signature scheme (Construction 4 in the lecture notes) is used as the underlying one-time signature scheme. (Where all messages are first hashed with a hash function $H$ before signing with Lamport’s scheme in order to fit in the message space.)

(a) Assume someone has implemented the signature scheme incorrectly as follows: Instead of using randomness from the pseudorandom function $F$ for the signing and key-generation algorithm, it runs signing and key-generation normally (i.e., as probabilistic algorithms, with fresh randomness each time it is invoked).

Explain how to break the signature scheme. More precisely, show how to sign an arbitrary message $m$ by performing only signature queries for messages $m' \neq m$. 

Note: Be explicit: describe all the actions and computations the adversary has to perform. (E.g., give the adversary in pseudocode.) It is not sufficient to say something like: “since two signatures are produced using the same key with a one-time signature scheme, the adversary can break the scheme”. Remember that the underlying scheme is Lamport’s one-time signature scheme.

(b) Bonus problem: Lamport’s signature scheme has public keys consisting of $2\eta \eta$-bit blocks (assuming that the one-way function $f$ has domain and range $\{0,1\}^\eta$). But it signs only messages of consisting of a single $\eta$-bit block. In the tree-based construction, we need to sign two Lamport public keys, i.e., $4\eta \eta$-bit blocks. Normally we solve this by converting Lamport’s scheme into a one-time signature scheme for long messages by hashing the messages to be signed.

Here we explore a different possibility. Instead of hashing the $4\eta \times \eta$ bits, we XOR the blocks together. That is, from Lamport’s scheme $(KG_{Lamport}, \text{Sign}_{Lamport}, \text{Verify}_{Lamport})$ we construct a one-time signature scheme $(KG_1, \text{Sign}_1, \text{Verify}_1)$ for $4\eta \times \eta$-bit messages as follows:

$$KG_1 := KG_{Lamport}.$$ \hspace{1cm} \text{Sign}_1(sk, m_1 || \ldots || m_{4\eta}) := \text{Sign}_{Lamport}(sk, \bigoplus_{i=1}^{4\eta} m_i) \text{ for } m_1, \ldots, m_{4\eta} \in \{0,1\}^\eta.$$ \hspace{1cm} \text{Verify}_1(pk, m_1 \ldots m_{4\eta}, \sigma) := \text{Verify}_{Lamport}(pk, \bigoplus_{i=1}^{4\eta} m_i, \sigma).

Now we can construct the tree-based signature scheme $(KG_{tree}, \text{Sign}_{tree}, \text{Verify}_{tree})$ from $(KG_1, \text{Sign}_1, \text{Verify}_1)$ without needing a hash function (as in Construction 7 in the lecture notes).

Your task: Break the resulting $(KG_{tree}, \text{Sign}_{tree}, \text{Verify}_{tree})$.

Note: It is not sufficient to just show that $(KG_1, \text{Sign}_1, \text{Verify}_1)$ is insecure. You have to break $(KG_{tree}, \text{Sign}_{tree}, \text{Verify}_{tree})$. All the other comments from the note of (a) also apply.