Problem 1: Tree-based signatures

This problem refers to the tree-based construction of signature schemes from one-time signatures from Construction 4 in the lecture notes. You may assume that Lamport’s signature scheme (Construction 2 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) As a warmup, let’s attack Lamport’s scheme. Assume Alice is sending random messages $m$, together with signatures $\sigma := \text{Sign}_{\text{Lamport}}(sk, m)$. Alice uses, though she should know better, the same $sk$ for all messages $m$. (I am not specifying how many messages Alice sends and signs, you can assume that there are enough of them for your attack.)

The adversary gets all messages $m$ and all corresponding signatures $\sigma$.

Describe how to efficiently compute $sk$ from the received $m, \sigma$.

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) Assume someone has implemented the signature scheme incorrectly as follows: Instead of using randomness from the pseudorandom function $F$ for the key-generation algorithm, it runs the 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.

(c) Bonus problem: Lamport’s signature scheme has public keys consisting of $2n$ $n$-bit blocks (assuming that the one-way function $f$ has domain and range $\{0, 1\}^n$). But it
signs only messages consisting of a single \( n \)-bit block. In the tree-based construction, we need to sign two Lamport public keys, i.e., \( 4n \) \( n \)-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 \( 4n \times n \) bits, we XOR the blocks together. That is, from Lamport’s scheme \((KG_{\text{Lamport}}, Sign_{\text{Lamport}}, Verify_{\text{Lamport}})\) we construct a one-time signature scheme \((KG_1, Sign_1, Verify_1)\) for \( 4n \times n \)-bit messages as follows:

\[
KG_1 := KG_{\text{Lamport}}. \quad Sign_1(sk, m_1 \parallel \ldots \parallel m_{4n}) := Sign_{\text{Lamport}}(sk, \bigoplus_{i=1}^{4n} m_i) \text{ for } m_1, \ldots, m_{4n} \in \{0, 1\}^n. \quad Verify_1(pk, m_1 \ldots m_{4n}, \sigma) := Verify_{\text{Lamport}}(pk, \bigoplus_{i=1}^{4n} m_i, \sigma).
\]

Now we can construct the tree-based signature scheme \((KG_{\text{tree}}, Sign_{\text{tree}}, Verify_{\text{tree}})\) from \((KG_1, Sign_1, Verify_1)\) without needing a hash function (as in Construction 4 in the lecture notes).

Your task: Break the resulting \((KG_{\text{tree}}, Sign_{\text{tree}}, Verify_{\text{tree}})\).

Note: It is not sufficient to just show that \((KG_1, Sign_1, Verify_1)\) is insecure. You have to break \((KG_{\text{tree}}, Sign_{\text{tree}}, Verify_{\text{tree}})\). All the other comments from the note of [5] also apply.