Question 1

P.S. SPHINCS is surprisingly annoying to type. I will abbreviate it with SPX.

We first briefly recall the key generation and signing procedure of SPX. In SPX key generation, a secret seed is randomly sampled, and a root WOTS keypair $(\text{PK}^{\text{WOTS}}_{\text{root}}, \text{SK}^{\text{WOTS}}_{\text{root}})$ is pseudorandomly generated from the secret seed. The SPX public key is the hash of the root WOTS public key, and the SPX secret key is the secret seed.

When signing a message, a FORS keypair $(\text{PK}^{\text{FORS}}, \text{SK}^{\text{FORS}})$ is pseudorandomly generated based on the message and the seed. The message is signed by FORS. The FORS public key is authenticated by a SPX hypertree (where each node is a XMDSS tree with 2^t WOTS keypairs). The SPX hypertree is finally authenticated using the root WOTS keypair by signing the root of the root XMSS tree in the SPX hypertree. The hyper tree is pseudorandomly generated using the secret seed alone so that under the same SPX keypair, the root WOTS keypair always signs the same value.

If the hyper tree is pseudorandomly generated using BOTH the secret seed and the message, then for distinct messages, each XMSS tree in the hyper tree will be distinct, and the root WOTS keypair will sign the root of a distinct XMSS tree, meaning that the root WOTS keypair signs distinct messages. **Therefore, with each distinct query signature query, the security of the root WOTS keypair degrades**.

A EF-CMA adversary

Let $\mathcal{A}$ denote an EF-CMA adversary against the modified SPX. It first makes a large number of distinct queries. For each query (m_i, σ_i), the returned signature σ_i contains a WOTS signature σ_i^{WOTS} signed using the root WOTS keypair. $\mathcal{A}$ chooses a message m^*, then executes the SPX.KeyGen() and SPX.Sign() routine up to the point where the root of the root XMSS tree of the hyper tree has been computed, which we denote by $\text{root}(T_0^*)$.

Because WOTS is a one-time signature, we know that with a large number of known signatures on distinct messages, the probability that $\mathcal{A}$ can forge a signature on $\text{root}(T_0^*)$ under the same keypair is non-negligible. Since $\mathcal{A}$ self-generated the FORS and the hypertree, all verification involving FORS and the hypertree will pass; if $\mathcal{A}$ can forge a signature under the true root WOTS keypair, then the entire verification will pass (we claim without proof that if $(\hat{m}, \hat{\sigma})$ is a valid forgery, then WOTS.PubKeyRegen$(\hat{m}, \hat{\sigma})$ will correctly recover the true PK^{WOTS}). Thus $\mathcal{A}$ has a non-negligible chance of successful forgery (and can quickly improve the probability of forgery by repeatedly sample different seeds).