TP : hash function and digital signatures

September 24, 2019

1 Merkle Tree

1.1 Preliminaries

Implement a function that takes two hashes as input and outputs the hash of their concatenation
Implement a function that takes a binary tree as input and outputs a boolean indicating whether this tree is a Merkle tree.

1.2 Complete Tree

Given a set of 2^n element implement a function that computes the Merkle tree of the set, a function that computes a witness for a given element, a function that checks a witness.

2 Public Key Infrastructure

2.1 Authority

Generate a pair of public private key and store them (this will be the keys of the authority)
Create a function that takes as input a public key and output the signature of the public key under the authority's key

2.2 Sub Authorities

Generates another key pair (this will be the sub authority's keys). Create a function that takes as input a public key, the authority secret key and outputs the sub authority public key, and the authority's signature of it.
Generate another key pair for a sub sub authority, and sign it with the sub authority.

2.3 User

Create a function that has harcoded the authority's public key, and takes as input a public key and a signature of the authority, and outputs a boolean indicating whether the key is signed by the authority
Create a function that has hardcoded the authority's public key and takes as input a public key signed by the sub auhtority, the sub auhtority public key, the auhtority's signature of the sub auhtority public key, and outputs a boolean indicating whether the key is valid

Create a function that has hardcoded the authority's public key and takes as input a public key signed by the sub sub authority, the necessary public keys and signatures and outputs a boolean indicating whether the key is valid

3 Incremental Merkle tree

We define an incremental Merkle tree of a set S of $m \leq 2^n$ element as the Merkle tree of S and a special element e copied $2^n - m$ times with e hashing to 0^{256}.
Create an incremental Merkle tree that uses space $O(m)$, can add an element in S, create a witness and check a witness in $O(n)$ time.
Hint : you only need to store the node modified by the elements of S and pre-compute the tree wih $S = \emptyset$