CSCI971 Advance Computer Security: Homework #6

Mei Wangzhihui
2019124044

Figure 1: Blockchain transaction verification in Merkle tree

Problem 1

Solution:

a) How does a Merkle tree work?

The original message are a sequence of l-bit blocks $x_1, x_2, ..., x_n$. We just verify several blocks which are used in in the block set to minimize the computation size. Each block can be verified independently.

The Merkle tree applied collision resistant function h to each block in $(x_1, x_2, ..., x_n) \in \mathcal{X}^n$ get the accordant hash value set $(y_1, y_2, ..., y_n) \in \mathcal{Y}^n$ by the algorithm:

for i = 1 to n, $y_i \leftarrow h(x_i)$

Then applied h to $(y_1, y_2, ..., y_n)$ to get the parental nodes $(y_{n+1}, ..., y_{2n+1})$ by the algorithm:

for i = 1 to n-1 , $y_{i+n} \leftarrow h(y_{2i-1}, y_{2i})$

So we get an binary tree with $2n + 1$ nodes.

When we want to verify the ith block ($\hat{x_i} = x_i?$), the algorithm need **Merkle proof** π, which is the intermediate hashes of siblings of nodes on the path from i to root. For example, set $i = 5, n = 8$, then $\pi = (y_6, y_{12}, y_{13})$.

We then do hash computation for $\hat{x_i}$ from ith node position to root. In this example, calculate:

$\hat{y_5} \leftarrow h(\hat{x_5})$

$\hat{y_{11}} = h(\hat{y_5}, y_6)$

$\hat{y_{15}} = h(\hat{y_{11}}, y_{13})$

If $\hat{y_{15}} = y_{15}$, then $hatx_5$ is verified, else not.

b) Why is it efficient when using Merkle tree to prove membership?

As the computation complexity for binary tree from leaf to root is $O(log_2^n)$, so the there need log_2^n times of hash computation in the verification process.

Also, Merkle tree algorithm need not secret keys.

c) How to take advantage of a Merkle tree to prove non-membership?

Suppose verifier want to verify that x is not in the list T, The verifier sort all the elements in T, and then build Merkle tree, this is sorted Merkle tree.

The verifier then find two adjancent leaves x_i, x_{i+1} and it satify that $x_i < x < x_{i+1}$. Next, verifier check Merkle proofs of x_i, x_{i+1}, to make sure x_i, x_{i+1} is in the Merkle tree. If $x \neq x_i$ and $x \neq x_{i+1}$ then x is not int the T, as x_i and x_{i+1} are adjancent leaf nodes. Else, x may be either x_i or x_{i+1}, so x is in the T.

d) How does blockchain use Merkle tree to verify transactions? Please describe by concrete example.

In the blockchain, each transaction need to be verified. The Merkle root maintain the integrity of all transaction data, which is stored in the block header. Each leaf node is hash of transactional data and non-leaf node is a hash of its previous hashes. As Merkle tree arebinary so it require even number of leaf nodes. If the number of nodes is odd, the last hash will be duplicated once to create an even number of leaf nodes.

We assume that transaction 8 happened in Figure **??** need verification. So verifier need not download the whole data but just the siblings hash nodes from its hash leaf to root, that is hash7, hash56, hash 1234, root hash. Apply hash computation with hash8 by the order, and the the transaction can be verified. The computation complexity is $O(log_2^n)$.