CS764_Homework_6

February 25, 2020

1 Blockchains and Cryptocurrencies

1.1 Homework 6

1.2 Joseph S Cannella

Q1. [5 points] Ethereum employs KECCAK256 hash algorithm to compute hashes in the modified MerklePatricia trie. Determine the KECCAK256 hash of the following root note of a Merkel tree. The hexa string to be hashed is “0B8FC549A” (Note: Here, all arehexa characters.) It is the same as “0b8fc549a” if you want to correlate with the notation in the below figure.These are NOT ASCII characters.

[150]:

```python
# Note: SHA3 is the official name of KECCAK 256
import hashlib, binascii
import math
hexInput = 0x0B8FC549A# hex values

# Much of what we need reiles on converting this to character array
# NOTE: the bytes will remain the same, jut the representation changes
def hexToString(hexVal):
    numBytes = math.ceil(math.log(hexVal, 256))
    prevByte = 0
    charArray = ""
    for i in range(numBytes, -1, -1):
        currentByte = hexVal>>(8*i)
        prevByte = currentByte<<(8*i)
        hexVal = hexVal - prevByte
        print(hex(currentByte))
        charArray += chr(currentByte)
    return charArray
# Calculate Character Array Representation of Hex String
chrArray = hexToString(0x0B8FC549A)
# Verification
inputStr = chr(0x0) + chr(0xB8) +  chr(0xFC) + chr(0x54) +  chr(0x9A)
if(chrArray == inputStr):
    print("Calcualtion Verified!")

# Compute Hash using SHA3
s = hashlib.sha3_256()
```

```
s.update(chrArray.encode())
print(f'Computed Hash: {s.hexdigest()}')
```

```
0x0
0xb8
0xfc
0x54
0x9a
Calcualtion Verified!
Computed Hash: 4fe3f0f1badb26168c66bd23ab36206fd90abd30a762564db07ce733e4830588
```

Q2. [15 points] As shown in the below figure, modified Merkle Patricia tries in Ethereumare used to store the world state. Here, the tree represents 4 given accounts (shown in the Simplified World State). Give the following 6 accounts, with account# being the key expressed as a hexa character string. For simplicity, account# is represented as a 8-character string. In reality, it is 40 characters or 20 bytes in length.

1. Construct a Merkle tree with these 6 accounts. Employ SHA-256 for hashing within the Merkel tree.

2. Construct a Patricia tree with these 6 accounts. Consider the address as a string of hexa characters. (iii)Construct a modified Merkle-Patricia tree (similar to the one in the below figure).

Account# (in hexa)	Account balance (in Ether)	Number of transactions
b35023b1	250.256	108
b57d46e8	4500.4798	213
b57690a1	367.90	578
d9a545b2	70013.256	1023
d9a7d235	678.23	651
d9a7d456	78.00	25

3. Compare the three implementations and comment why Ethereum inventor proposed the modified Merkle-Patricia tree. First, create a transaction class for the individual transactions.

[148]:

```
# Transactions Class
class Transaction:
    account = ""
    balance = 0
    numTrans = 0
    def __init__(self, account, balance, numTrans):
        self.account = account
        self.balance = balance
        self.numTrans = numTrans
```

Now that we have created our basic classes we begin by constructing a list of transactions.

[149]:
```
transactions = []
transactions.append(Transaction(0xb35023b1, 250.256, 108))
transactions.append(Transaction(0xb57d46e8, 4500.4798, 213))
transactions.append(Transaction(0xb57690a1, 367.90, 578))
transactions.append(Transaction(0xd9a545b2, 70013.256, 1023))
transactions.append(Transaction(0xd9a7d235, 678.23, 651))
transactions.append(Transaction(0xd9a7d456, 78.00, 25))

for t in transactions:
    print(f'account: {t.account}, Balance: {t.balance}, # Transactions {t.
 ↪numTrans}')
```

```
account: 3008373681, Balance: 250.256, # Transactions 108
account: 3044886248, Balance: 4500.4798, # Transactions 213
account: 3044446369, Balance: 367.9, # Transactions 578
account: 3651487154, Balance: 70013.256, # Transactions 1023
account: 3651654197, Balance: 678.23, # Transactions 651
account: 3651654742, Balance: 78.0, # Transactions 25
```

(i) Construct Merkle Tree implementing SHA-256

[ ]:
```

```

(iii) Construct a modified Merkle-Patricia Tree

Approach We require three types of nodes to build modified Merkle Patricia Tree 1. Leaf Nodes - containing the actual value of a transaction 2. Extension Nodes - 3. Branch Nodes - basically a 16 elemennt array or pointers to children nodes

[ ]:
```
# Leaf Node Class
class LeafNode:
    prefix = 0
    keyEnd = 0
    value = 0
    def __init__(self, prefix, keyEnd, value):
        self.prefix = prefix
        self.keyEnd = keyEnd
        self.value = value

# Extension Node Class
class ExtensionNode:
    prefix = 0
    sharedNibles = 0
    def __init__(self, prefix, sharedNibbles, nextNode):
        self.prefix = prefix
        self.sharedNibbles = sharedNibbles
        self.nextNode = nextNode

# Branch Node Class
class BranchNode:
    address = 0
    value = 0
    def __init__(self, addresses, value):
        self.addresses = addresses
        self.value = value
```