
Verification in Hybrid Merkle Tree

```
bool verifyHybrid(string t, int pos, vector<string> &trans, unordered_map<int, vector<string>> mp, string &root){
string hash = sha256(t);
hashCount = 0;
int size = trans.size(); int levels = ((ceil(log3(size, 3))) + 1); int i = 2;
while (i <= levels){
int prevSz = mp[i - 1].size(); int prevPos = pos;
if (prevPos % 3 == 1){
string h1 = hash;
string h2 = "", h3 = "";
if (prevPos + 1 <= mp[i - 1].size())
h2 = mp[i - 1][prevPos];
if (prevPos + 2 <= mp[i - 1].size())
h3 = mp[i - 1][prevPos + 1];
reverse(h1.begin(), h1.end()); reverse(h2.begin(), h2.end()); reverse(h3.begin(), h3.end());
hashCount += 3;
string str = "";
str += h3;
str += h2;
str += h1;
hash = sha256(str);
pos = (prevPos / 3) + (prevPos % 3 != 0 ? 1 : 0);
}
else if (prevPos % 3 == 2){
string h1 = mp[i - 1][prevPos - 2]; string h2 = hash; string h3 = "";
if (prevPos + 1 <= mp[i - 1].size())
h3 = mp[i - 1][prevPos];
reverse(h1.begin(), h1.end()); reverse(h2.begin(), h2.end()); reverse(h3.begin(), h3.end());
hashCount += 3;
string str = "";
str += h3;
str += h2;
str += h1;
hash = sha256(str);
pos = (prevPos / 3) + (prevPos % 3 != 0 ? 1 : 0);
}
else if (prevPos % 3 == 0){
string h1 = mp[i - 1][prevPos - 3]; string h2 = mp[i - 1][prevPos - 2]; string h3 = hash;
reverse(h1.begin(), h1.end()); reverse(h2.begin(), h2.end()); reverse(h3.begin(), h3.end());
hashCount += 3;
string str = "";
str += h3;
str += h2;
str += h1;
hash = sha256(str);
pos = (prevPos / 3) + (prevPos % 3 != 0 ? 1 : 0);
}
i++;
}
if (hash == root)
return true;
return false;
}
```
