POIS : Evaluation 3

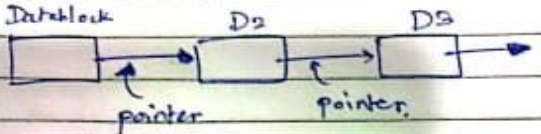
Rishabh Singhal - 20171213

Date :__/__/__

POIS EVAL 3

Rishabh Singhal
20171213

Ans - Q) For this ~~take~~ task, I'll implement linked list.
Where linked list :-



here in each datablock, it will store the address of the next datablock + hash of that datablock (for Hash) & pointer

(Hash + Sign) for 3rd implementation.

Advantages of hash :-

~~Pointers~~ Pointers will only store the address, while hash pointer will store hash too, so if there is any change in the next block it can be detected by checking the hash of next block & previously stored hash.

~~Also, these~~ this exact datastructure is ~~also~~ used in blockchain & cryptocurrency.

~~For implementation, it~~

☆ Advantage of hash-sign :-

Here we can verify the person (~~who~~ who stored data in the data block) with the public key also, + P+ have all the advantages of (hash-~~to~~ pointer) implementation.

↳ This can be applied in online transactions (banking) & digital certificates.

Implementation Details

1. Linked List :
    a. For this, 2 classes are implemented:
        i. LinkedList
            1. Methods
                a. __init__ : to initialize LinkedList headPointer
                b. appendFront(data) : to add “data” block to front
                c. appendEnd(data) : to add “data” block at the end
                d. traverseList: to traverse List
            2. Variables:
                a. hpointer_next: pointer to the head
                b. address: hash-map(dictionary) for {key:pointer, value:block}
        ii. Block:
            1. Methods
                a. __init__: to initialize block with data, and pointer to next block
                b. __repr__: for printing block in a string format
            2. Variables
                a. Data
                b. Pointer_next: pointer to the next block
                c. addkey: address to this block
2. Hashed Link List:
    a. For this, 2 classes are implemented:
        i. HashLinkedList
            1. Methods
                a. __init__ : to initialize Hashed LinkedList headPointer, and hash of next block
                b. appendBlock(data) : to add “data” block to front and calculating hash
                c. traverseList: to traverse List and simultaneously check if the block hash matches with the already stored one.
            2. Variables:
                a. hpointer_next: pointer to the head
                b. address: hash-map(dictionary) for {key:pointer, value:block}
                c. hhash_next: hash of the head
        ii. HashBlock
            1. Methods
                a. __init__: to initialize block with data, and pointer to next block
                b. __repr__: for printing block in a string format
                c. calc_hash: to calculate hash of the present state of block
            2. Variables
                a. Data
                b. pointer_next: pointer to the next block
                c. hash_next: hash of the next block (pointed to)
                d. addkey: address to this block
3. Sign Hash Link List:
    a. For this, 2 classes are implemented:
        i. SignHashLinkedList
            1. Methods

a. __init__ : to initialize Hashed LinkedList headPointer, and hash of next block
b. appendBlock(data) : to add “data” block to front and calculating hash
c. traverseList: to traverse List and simultaneously check if the block signature verifies or not.

2. Variables:
a. hpointer_next: pointer to the head
b. address: hash-map(dictionary) for {key:pointer, value:block}
c. hhash_next: hash of the head
d. hsign_next: signature of the hash of the head
e. pid_next: person id of the person who created head block

ii. SignHashBlock

1. Methods
a. __init__: to initialize block with data, and pointer to next block
b. __repr__: for printing block in a string format
c. calc_hash: to calculate hash of the present state of block

2. Variables
a. Data
b. pointer_next: pointer to the next block
c. hash_next: hash of the next block (pointed to)
d. Sign_next: sign of the next block
e. pid_next: person id of the person who created next block
f. addkey: address to this block

4. Person Class

Person Object stores the private key and public key, which is generated for the first time, when it is created.

Use cases are discussed in the first page. :)