searchElement.py

Description: This module will take the bloom filter of the node, the vr of that node and also the trap value as input and generates the second hash value by appending vr to each element in trap and hashing it, then it mods it with the size of the bloom filter which will give it the index in the bloom filter. Then it checks if that index in bloom is 1, which means that element is found. If the index has a value 0, that means that the element is not present at that node and therefore, stops searching further.

Input: bloom (Bloom Filter of the node), vr (Private key of the node), trap (One time hashed values of that node) and few other for testing purpose which are unnecessary for execution

Complexity: $O(n^2)$ (Here, n = number of elements in the trap)

Output: Returns **PASS** if the element is found in the bloom filter.

Code:

```
import sys
import hashlib # Used to generate Hash values

# SHA-1 Hash function!
def hashIt(s):
    s = str.encode(s) # converts the string to bytes
    return hashlib.sha1(s).hexdigest() # Returns the hashed value

# This is the main function which takes BloomFilter, Private Key of nodes
# Trap value of the node as input and searches for the indices of bloom for
# a value = 1, if present then it returns PASS which will notify the caller
# that the element has been found in the respective bloomFilter
def searchForME(bloom, vr, secondhashed, trap, SNUM):
    trapCopy = trap[:]
    m = len(bloom) # BloomFilter length

    # Checks this for every element in the trap (7 * numOfUniquePrefixes of all the elements in the
node)
    for row in trap:
        for col in row:
            twoTimeHash = hashIt(col + str(vr)) # This appends the vr to the trap and hashes it
            # print(twoTimeHash) # Prints the hash with vr
            xa = int(twoTimeHash, 16) # Converts the hashed value to decimal
            index = xa % m # Gives an index in the bloomFilter
            if bloom[index] == 1: # If that index in bloom == 1, then element found!
                return "PASS"
            else:
                break # Element not found, stop the search
```