Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Experiment 7

Date of Performance : 27 March 2023

Date of Submission: 27 March 2023

SAP Id: 60004200132

Name : Ayush Jain

Div: B

Batch : B3

Aim of Experiment

Implement Merkle Root creation with the help of SHA-256. Your program will have input as paragraph. Paragraph can be converted to suitable blocks for which hash values can be computed. Finally generate Merkle root based on these computed hash values.

Theory / Algorithm / Conceptual Description:

SHA-256:

SHA-256 is an algorithm used for hash functions and is a vital component of contemporary cybersecurity. It is part of the Secure Hash Algorithm 2 (SHA-2), which was created by the National Security Agency (NSA) in 2001. The name SHA-256 refers to the 256-bit long output value of the hash function.

SHA-256 can best be understood as a collection of cryptographic hash functions. A hash function, also referred to as digest or fingerprint, is like a unique signature for a data file or text. It cannot be read or decrypted, as it only allows for a one-way cryptographic function. This allows hashing to be used for the verification of files, digital signatures, secure messages, and other applications.

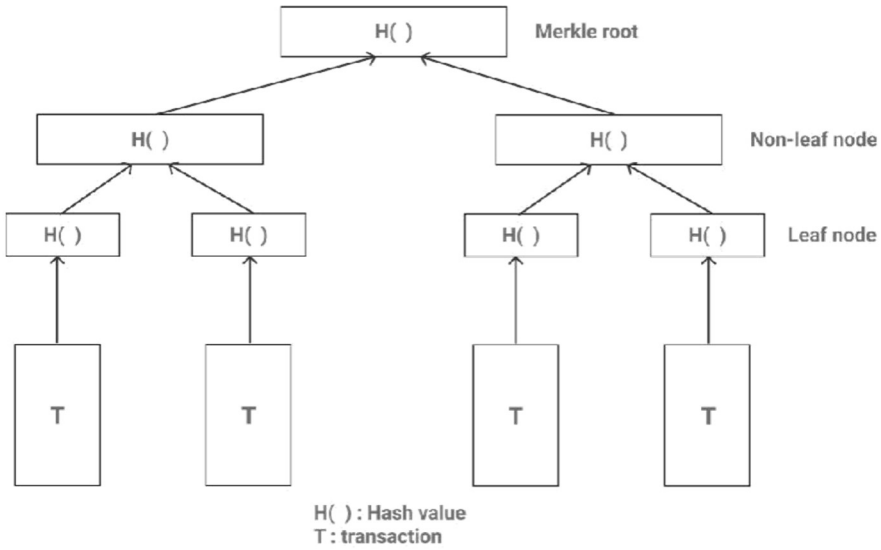
There are several steps involved in using SHA-256, including pre-processing a file or text into binary, initializing hash values and round constants, creating a message schedule, compressing, modifying final values, and completing a string concatenation to get all the bits together.

Merkle Root:

A hash tree is also known as Merkle Tree. It is a tree in which each leaf node is labeled with the hash value of a data block and each non-leaf node is labeled with the hash value of its child nodes labels.

A Merkle tree is constructed from the leaf nodes level all the way up to the Merkle root level by grouping nodes in pairs and calculating the hash of each pair of nodes in that particular level. This hash value is propagated to the next level. This is a bottom-to-up type of construction where the hash values are flowing from down to up direction.

The following diagram shows the structure of a Merkle Tree:



Steps involved:

1) The hash of each transaction is computed.
   H1 = Hash(T1).

2) The hashes computed are stored in leaf nodes of the Merkle tree.

3) Now non-leaf nodes will be formed. In order to form these nodes, leaf nodes will be paired together from left to right, and the hash of these pairs will be calculated. Firstly, hash of H1 and H2 will be computed to form H12. Similarly, H34 is computed.
   Values H12 and H34 are parent nodes of H1, H2, and H3, H4 respectively. These are non-leaf nodes.

   H12 = Hash(H1 + H2)
H34 = Hash(H3 + H4)

4) Finally, H1234 is computed by pairing H12 and H34. H1234 is the only hash remaining.
   This means we have reached the root node and therefore H1234 is the Merkle root. H1234 = Hash(H12 + H34)

Program

```
from hashlib import sha256
import math
def hash(x):
  ans=sha256(x.encode("utf-8")).hexdigest()
  return ans
def combine(l):
  temp=[]
  n=len(l)
  i=0
  while i<n-1:
    temp.append(l[i]+l[i+1])
    i=i+2
    return temp

para="hello\ngoodbye\nblue\ncrystal\npuppy\nsandalwood\nlamp\nfineprint"
para1="hello\ngoodbye\nblue\ncrystal\npuppy\nsandalwood\nlamp"
para2="hello\ngoodbye\nblue\ncrystal\npuppy\nsandalwood\nlamp\nfineprint
\nmyrtle"

l=[]
n=para2.count("\n")+1
for i in range(n):
  l.append("")
pos=0
for i in para2:
  if i=="\n":
    pos=pos+1
    continue
  l[pos]=l[pos]+i

while len(l)<8:
  l.append(l[-1])

if len(l)>8:
  p=math.log(n,2)
  diff=p-int(p)
  if diff!=0:
    p=int(p+1)
  while len(l)!=2**p:
    l.append(l[-1])
while len(l)!=1:
  hashval=[]
```

```
  for i in l:
    hashval.append(hash(i))
  l=combine(hashval)

merkleroot=hash(l[0])
print("Merkle Root: ",merkleroot)
```

Input

Case 1:

```
para="hello\ngoodbye\nblue\ncrystal\npuppy\nsandalwood\nlamp\nfineprint"
```

Case 2:

```
para1="hello\ngoodbye\nblue\ncrystal\npuppy\nsandalwood\nlamp"
```

Case 3:

```
para2="hello\ngoodbye\nblue\ncrystal\npuppy\nsandalwood\nlamp\nfineprint
\nmyrtle"
```

Output

Case 1:

```
Merkle Root:  d035aa4edb501691c46b6d0cc11a42502e50fb63735d4535a0fd4b6149aca276
```

Case 2:

```
Merkle Root:  dbc2a1e6c0eec7601eaedc7df58214c579aa53dbd7727e7d38f13be2d308ecd6
```

Case 3:

```
Merkle Root:  7e20320f4c93b54b92722d30867aa7bc0f5e7f6ebc5037a90ecdfa5b4a1ceeec
```

CONCLUSION: Thus, we have designed and implemented Merkle Root creation with the help of SHA-256.