**MD5 HASHING ALGORITHM**

**BY-**

**AAKRITI MEHROTRA**

**22BCE1954**

**CODE**

**SERVER.PY**

import socket

import struct

from math import floor, sin

from bitarray import bitarray

# MD5 Constants

MD5\_BUFFER = {

'A': 0x67452301,

'B': 0xEFCDAB89,

'C': 0x98BADCFE,

'D': 0x10325476,

}

# MD5 Functions

def F(x, y, z):

return (x & y) | (~x & z)

def G(x, y, z):

return (x & z) | (y & ~z)

def H(x, y, z):

return x ^ y ^ z

def I(x, y, z):

return y ^ (x | ~z)

def rotate\_left(x, n):

return (x << n) | (x >> (32 - n))

def modular\_add(a, b):

return (a + b) % pow(2, 32)

def md5\_hash(message):

# Step 1: Pad the message

bit\_array = bitarray(endian="big")

bit\_array.frombytes(message.encode("utf-8"))

bit\_array.append(1)

while len(bit\_array) % 512 != 448:

bit\_array.append(0)

# Step 2: Append the length

length = (len(message) \* 8) % pow(2, 64)

length\_bit\_array = bitarray(endian="little")

length\_bit\_array.frombytes(struct.pack("<Q", length))

bit\_array.extend(length\_bit\_array)

# Step 3: Initialize buffers

buffers = MD5\_BUFFER.copy()

# Step 4: Process chunks

T = [floor(pow(2, 32) \* abs(sin(i + 1))) for i in range(64)]

N = len(bit\_array) // 32

for chunk\_index in range(N // 16):

start = chunk\_index \* 512

X = [bit\_array[start + (x \* 32): start + (x \* 32) + 32] for x in range(16)]

X = [int.from\_bytes(word.tobytes(), byteorder="little") for word in X]

A, B, C, D = buffers['A'], buffers['B'], buffers['C'], buffers['D']

for i in range(64):

if 0 <= i <= 15:

k = i

s = [7, 12, 17, 22]

temp = F(B, C, D)

elif 16 <= i <= 31:

k = ((5 \* i) + 1) % 16

s = [5, 9, 14, 20]

temp = G(B, C, D)

elif 32 <= i <= 47:

k = ((3 \* i) + 5) % 16

s = [4, 11, 16, 23]

temp = H(B, C, D)

elif 48 <= i <= 63:

k = (7 \* i) % 16

s = [6, 10, 15, 21]

temp = I(B, C, D)

temp = modular\_add(temp, X[k])

temp = modular\_add(temp, T[i])

temp = modular\_add(temp, A)

temp = rotate\_left(temp, s[i % 4])

temp = modular\_add(temp, B)

A, B, C, D = D, temp, B, C

buffers['A'] = modular\_add(buffers['A'], A)

buffers['B'] = modular\_add(buffers['B'], B)

buffers['C'] = modular\_add(buffers['C'], C)

buffers['D'] = modular\_add(buffers['D'], D)

# Step 5: Output the hash

A = struct.unpack("<I", struct.pack(">I", buffers['A']))[0]

B = struct.unpack("<I", struct.pack(">I", buffers['B']))[0]

C = struct.unpack("<I", struct.pack(">I", buffers['C']))[0]

D = struct.unpack("<I", struct.pack(">I", buffers['D']))[0]

return f"{format(A, '08x')}{format(B, '08x')}{format(C, '08x')}{format(D, '08x')}"

# Server Setup

def start\_server():

server\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server\_socket.bind(('localhost', 12345))

server\_socket.listen(1)

print("Server is listening on port 12345...")

while True:

client\_socket, addr = server\_socket.accept()

print(f"Connection from {addr} established.")

message = client\_socket.recv(1024).decode('utf-8')

print(f"Received message: {message}")

hashed\_message = md5\_hash(message)

print(f"Hashed message: {hashed\_message}")

client\_socket.send(hashed\_message.encode('utf-8'))

client\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_":

start\_server()

**CLIENT.PY**

import socket

def start\_client():

client\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client\_socket.connect(('localhost', 12345))

message = input("Enter a message to hash: ")

client\_socket.send(message.encode('utf-8'))

hashed\_message = client\_socket.recv(1024).decode('utf-8')

print(f"Hashed message from server: {hashed\_message}")

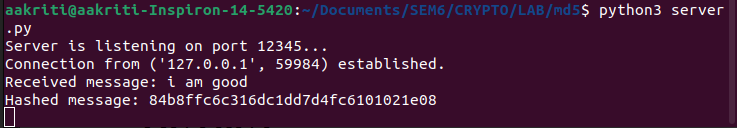
client\_socket.close()

if \_\_name\_\_ == "\_\_main\_\_":

start\_client()

**OUTPUT**

**SERVER**

****

**CLIENT**

