**SHA-128**

**BY-**

**Aakriti Mehrotra**

**22BCE1954**

**CODE**

**SERVER**

import socket

# Import the SHA-128 function

from impl import sha128

HOST = '127.0.0.1' # Localhost

PORT = 65432 # Port to listen on

def start\_server():

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as server\_socket:

server\_socket.bind((HOST, PORT))

server\_socket.listen()

print(f"Server listening on {HOST}:{PORT}...")

while True:

conn, addr = server\_socket.accept()

with conn:

print(f"Connected by {addr}")

data = conn.recv(1024)

if not data:

break

message = data.decode()

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

# Compute SHA-128 hash

hashed\_message = sha128(message)

print(f"Sending hashed output: {hashed\_message}")

# Send back the hash

conn.sendall(hashed\_message.encode())

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

start\_server()

**CLIENT**

import socket

HOST = '127.0.0.1' # Localhost (same as server)

PORT = 65432 # Same port as server

def send\_message(message):

with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as client\_socket:

client\_socket.connect((HOST, PORT))

client\_socket.sendall(message.encode())

# Receive response from the server

hashed\_response = client\_socket.recv(1024).decode()

print(f"SHA-128 Hash: {hashed\_response}")

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

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

send\_message(message)

**IMPLEMENTATION**

import struct

import math

def sha128(message):

if isinstance(message, str):

message = message.encode('utf-8')

h0 = 0x6a09e667

h1 = 0xbb67ae85

h2 = 0x3c6ef372

h3 = 0xa54ff53a

k = [

0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5,

0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,

0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,

0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174

]

original\_bit\_length = len(message) \* 8

message += b'\x80'

while (len(message) % 64) != 56:

message += b'\x00'

message += struct.pack('>Q', original\_bit\_length)

for chunk\_start in range(0, len(message), 64):

chunk = message[chunk\_start:chunk\_start + 64]

w = [0] \* 64

for i in range(16):

w[i] = struct.unpack('>I', chunk[i\*4:i\*4+4])[0]

for i in range(16, 64):

s0 = rightrotate(w[i-15], 7) ^ rightrotate(w[i-15], 18) ^ (w[i-15] >> 3)

s1 = rightrotate(w[i-2], 17) ^ rightrotate(w[i-2], 19) ^ (w[i-2] >> 10)

w[i] = (w[i-16] + s0 + w[i-7] + s1) & 0xFFFFFFFF

a, b, c, d = h0, h1, h2, h3

for i in range(64):

if i < 16:

s1 = rightrotate(a, 2) ^ rightrotate(a, 13) ^ rightrotate(a, 22)

ch = (a & b) ^ ((~a) & c)

temp1 = (d + s1 + ch + k[i % 16] + w[i]) & 0xFFFFFFFF

s0 = rightrotate(a, 6) ^ rightrotate(a, 11) ^ rightrotate(a, 25)

maj = (a & b) ^ (a & c) ^ (b & c)

temp2 = (s0 + maj) & 0xFFFFFFFF

d, c, b, a = c, b, a, (temp1 + temp2) & 0xFFFFFFFF

h0 = (h0 + a) & 0xFFFFFFFF

h1 = (h1 + b) & 0xFFFFFFFF

h2 = (h2 + c) & 0xFFFFFFFF

h3 = (h3 + d) & 0xFFFFFFFF

return '%08x%08x%08x%08x' % (h0, h1, h2, h3)

def rightrotate(value, shift):

return ((value >> shift) | (value << (32 - shift))) & 0xFFFFFFFF

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

test\_strings = [

"",

"a",

"abc",

"message digest",

"abcdefghijklmnopqrstuvwxyz",

"The quick brown fox jumps over the lazy dog"

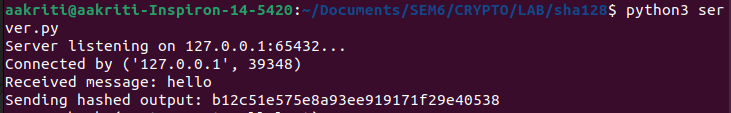
]

for s in test\_strings:

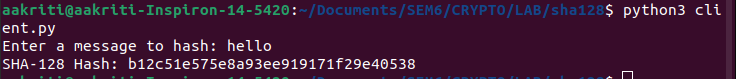
print(f'SHA-128("{s}") = {sha128(s)}')

**OUTPUT**

**SERVER**

****

**CLIENT**

****