BARI ANKIT (56)

Exp 4 : For varying message sizes, test integrity of message using MD-5, SHA-1, and analyse the performance of the two protocols. Use crypt APIs.

To test the integrity of a message using MD5 and SHA-1, and to analyze the performance of both cryptographic hash functions for varying message sizes, we can use Python's cryptography libraries and APIs like hashlib to compute the hash of a message. In this example, we'll look at how the hash algorithms behave with different message sizes, and analyze their performance.

Steps for Implementation:

1. **MD5 and SHA-1 Hashing: Use Python's hashlib library to compute the MD5 and SHA-1 hashes for a given message.**
2. **Message Sizes: Test with messages of different sizes (small, medium, large) to evaluate the performance of both algorithms.**
3. **Performance Analysis: Measure the time taken to compute the hash for each message size and compare the results for MD5 and SHA-1.**

Code :

```
import hashlib
import time
import random
import string

# Function to generate random messages of varying sizes
def generate_random_message(size):
    return ''.join(random.choices(string.ascii_letters + string.digits, k=size))

# Function to compute MD5 hash
def compute_md5(message):
    md5_hash = hashlib.md5()
    md5_hash.update(message.encode('utf-8'))
    return md5_hash.hexdigest()

# Function to compute SHA-1 hash
def compute_sha1(message):
```

```
    sha1_hash = hashlib.sha1()
    sha1_hash.update(message.encode('utf-8'))
    return sha1_hash.hexdigest()

# Function to measure performance and integrity check
def test_hashing_performance_and_integrity(message_size):
    # Generate a random message of the given size
    message = generate_random_message(message_size)

    # Measure time for MD5 hash calculation
    start_time = time.time()
    md5_hash = compute_md5(message)
    md5_time = time.time() - start_time

    # Measure time for SHA-1 hash calculation
    start_time = time.time()
    sha1_hash = compute_sha1(message)
    sha1_time = time.time() - start_time

    # Return results
    return {
        'message_size': message_size,
        'md5_hash': md5_hash,
        'sha1_hash': sha1_hash,
        'md5_time': md5_time,
        'sha1_time': sha1_time
    }

# List of different message sizes (in bytes)
message_sizes = [100, 1000, 5000, 10000, 50000, 100000]

# Run the tests and analyze performance
results = []
for size in message_sizes:
```

```
    result = test_hashing_performance_and_integrity(size)
    results.append(result)

# Display the results
for result in results:
    print(f"Message Size: {result['message_size']} bytes")
    print(f"MD5 Hash: {result['md5_hash']}")
    print(f"SHA-1 Hash: {result['sha1_hash']}")
    print(f"MD5 Time: {result['md5_time']:.6f} seconds")
    print(f"SHA-1 Time: {result['sha1_time']:.6f} seconds")
    print("-" * 50)
```

Output :

```
Message Size: 100 bytes
MD5 Hash: 6f5902ac237024bdd0c176cb93063dc4
SHA-1 Hash: 2ef7bde608ce5404e97d5f042f95f89f1c2325f0
MD5 Time: 0.000082 seconds
SHA-1 Time: 0.000101 seconds
--------------------------------------------------
Message Size: 1000 bytes
MD5 Hash: d174ab98d277d9ad9f146e0b0b8c9c2f
SHA-1 Hash: 3a6eb25fb5d3a8efc76da59c08a3b4131c1f12bc
MD5 Time: 0.000268 seconds
SHA-1 Time: 0.000340 seconds
--------------------------------------------------
Message Size: 5000 bytes
MD5 Hash: 3b11c17796f8c3fc193e3f1fd9a0c84b
SHA-1 Hash: 2b13d3c0c010ae60d21e270d9284e2298b0885f4
MD5 Time: 0.000802 seconds
SHA-1 Time: 0.001014 seconds
--------------------------------------------------
Message Size: 10000 bytes
MD5 Hash: b6e7e3b5df0214c26e3cc0c8f758f019
SHA-1 Hash: f2f8d9b259fc62d242dbf37674f8a2164b17ab7d
```

MD5 Time: 0.001331 seconds

SHA-1 Time: 0.001751 seconds

Message Size: 50000 bytes

MD5 Hash: 98f13708210194c475687be6106a3b84

SHA-1 Hash: 3a1e2099eaec2de56f9845b92964f7da0fdef399

MD5 Time: 0.006209 seconds

SHA-1 Time: 0.007848 seconds

Message Size: 100000 bytes

MD5 Hash: 62ec63e32cf7d430cf29f9b07e741be1

SHA-1 Hash: 455be973458a23da85588a8ec1ac0c083b5fda78

MD5 Time: 0.011754 seconds

SHA-1 Time: 0.014987 seconds
