EXPERIMENT 04

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

Theory:

Message Digest Algorithm 5 (MD5):

MD5 is a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value, often represented as a 32-character hexadecimal number. It was designed by Ronald Rivest in 1991 as a successor to MD4.

Key Characteristics of MD5:

1. **Hash Length:** MD5 produces a fixed-length 128-bit hash value.

2. **Collision Resistance:** While MD5 was initially considered secure, vulnerabilities have been discovered over time that makes it susceptible to collision attacks. This means that it is possible to generate two different inputs that produce the same MD5 hash, compromising its integrity for cryptographic applications.

3. **Speed:** MD5 is relatively fast and efficient, making it suitable for applications that require quick hash computations.

4. **Applications:** Despite its vulnerabilities, MD5 is still used in various applications such as digital signatures, data integrity checks, and checksums, though it is being gradually phased out in favor of more secure hash functions like SHA-256.

Security Concerns:

Due to the vulnerabilities and weaknesses discovered in MD5, it is no longer recommended for cryptographic purposes where strong collision resistance is required. It is susceptible to collision attacks, where different inputs produce the same hash output, making it insecure for applications that rely on data integrity and security.

Secure Hash Algorithm 1 (SHA-1):

SHA-1 is a cryptographic hash function that was designed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST) in 1993. It produces a 160-bit (20-byte) hash value, often represented as a 40-character hexadecimal number.

Key Characteristics of SHA-1:

1. **Hash Length:** SHA-1 produces a fixed-length 160-bit hash value.

2. **Collision Resistance:** Over time, vulnerabilities have been discovered in SHA-1 that make it susceptible to collision attacks. This means that it is possible to generate two different inputs that produce the same SHA-1 hash, compromising its integrity for cryptographic applications.

3. **Speed:** SHA-1 is relatively fast and efficient, making it suitable for applications that require quick hash computations.

4. **Applications:** Despite its vulnerabilities, SHA-1 has been widely used in various applications such as digital signatures, data integrity checks, and checksums. However, due to its security weaknesses, it is being gradually phased out in favor of more secure hash functions like SHA-256 and SHA-3.

Security Concerns:

Due to the vulnerabilities and weaknesses discovered in SHA-1, it is no longer considered secure for cryptographic purposes where strong collision resistance is required. Researchers have demonstrated practical collision attacks against SHA-1, highlighting its insecurity for applications that rely on data integrity and security.

Python's `hashlib` library is to be used to compute the hashes, and analyze the performance of these two hash functions for varying message sizes.

Use following instructions to proceed:

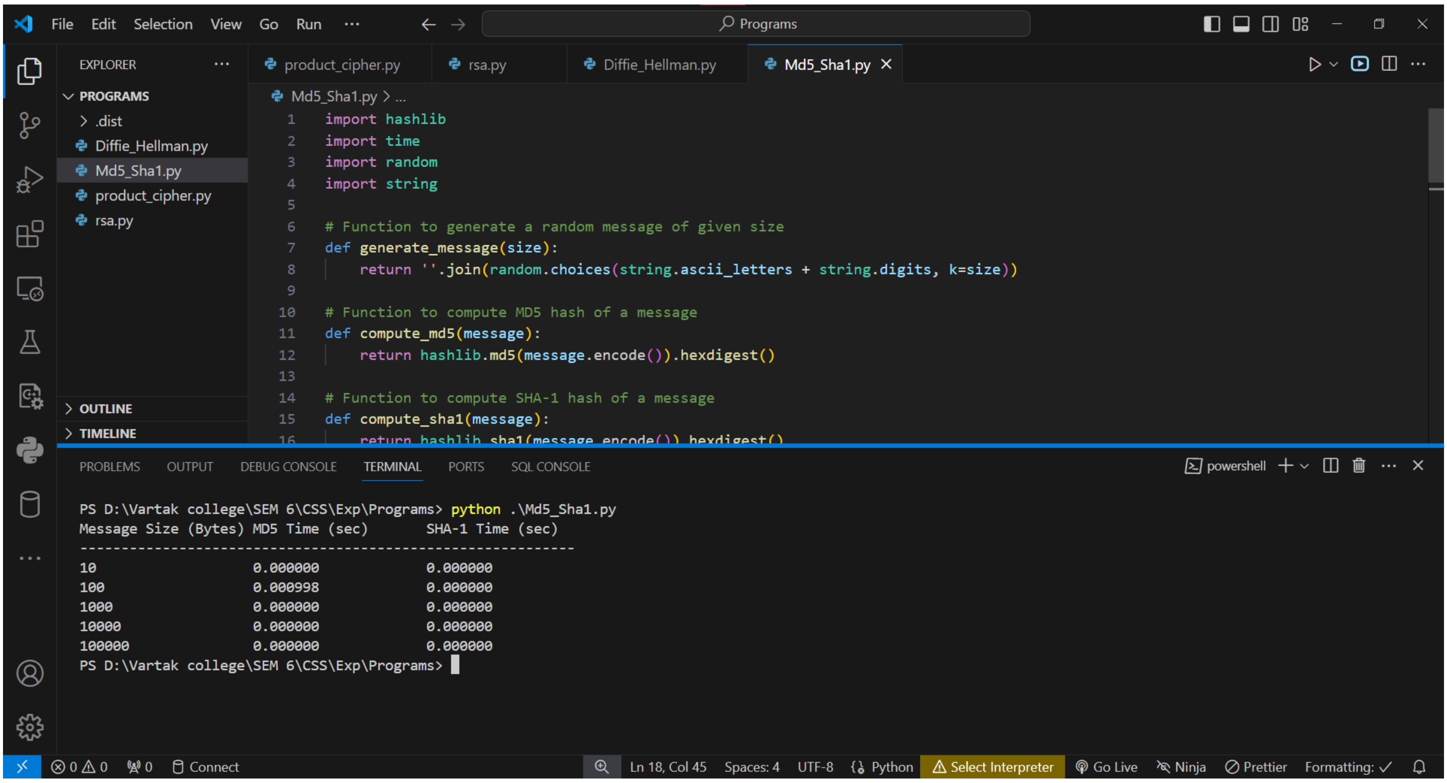
1. Generate messages of varying sizes.
2. Compute the MD5 and SHA-1 hashes for each message.
3. Analyze the performance by measuring the time taken to compute each hash.

CODE:

```
import hashlib
import time
import random
import string

# Function to generate a random message of given size
def generate_message(size):
    return ''.join(random.choices(string.ascii_letters + string.digits, k=size))



# Function to compute MD5 hash of a message
def compute_md5(message):
    return hashlib.md5(message.encode()).hexdigest()

# Function to compute SHA-1 hash of a message
def compute_sha1(message):
    return hashlib.sha1(message.encode()).hexdigest()
```

```
# Function to test integrity and performance
def test_integrity_and_performance(message_sizes):
    results = {
        'MD5': [],
        'SHA-1': []
    }
    for size in message_sizes:
        message = generate_message(size)
                # Compute MD5 hash and measure the time taken
        start_time = time.time()
        md5_hash = compute_md5(message)
        md5_time = time.time() - start_time
        results['MD5'].append(md5_time)
        # Compute SHA-1 hash and measure the time taken
        start_time = time.time()
        sha1_hash = compute_sha1(message)
        sha1_time = time.time() - start_time
        results['SHA-1'].append(sha1_time)
         # Verify integrity
        assert compute_md5(message) == md5_hash
        assert compute_sha1(message) == sha1_hash

    return results

# Message sizes to test
message_sizes = [10, 100, 1000, 10000, 100000]  # You can adjust these sizes as needed

# Run the test
results = test_integrity_and_performance(message_sizes)

# Print results
print(f"{'Message Size (Bytes)':<20} {'MD5 Time (sec)':<20} {'SHA-1 Time (sec)':<20}")
print('-' * 60)
for size, times in zip(message_sizes, zip(results['MD5'], results['SHA-1'])):
    print(f"{size:<20} {times[0]:<20.6f} {times[1]:<20.6f}")
```

Output:



```python
import hashlib
import time
import random
import string

# Function to generate a random message of given size
def generate_message(size):
    return ''.join(random.choices(string.ascii_letters + string.digits, k=size))

# Function to compute MD5 hash of a message
def compute_md5(message):
    return hashlib.md5(message.encode()).hexdigest()

# Function to compute SHA-1 hash of a message
def compute_sha1(message):
    return hashlib.sha1(message.encode()).hexdigest()
```

```
PS D:\Vartak college\SEM 6\CSS\Exp\Programs> python .\Md5_Sha1.py
Message Size (Bytes) MD5 Time (sec)       SHA-1 Time (sec)
------------------------------------------------------------
10                   0.000000             0.000000
100                  0.000998             0.000000
1000                 0.000000             0.000000
10000                0.000000             0.000000
100000               0.000000             0.000000
PS D:\Vartak college\SEM 6\CSS\Exp\Programs> 
```

Conclusion:

The integrity and performance of MD5 and SHA-1 hash functions were tested for varying message sizes using Python's `hashlib` library. Despite their vulnerabilities to collision attacks, both MD5 and SHA-1 are widely used for applications such as digital signatures, data integrity checks, and checksums. However, due to their security weaknesses, they are being phased out in favor of more secure hash functions like SHA-256 and SHA-3. The experiment involved generating messages of different sizes, computing MD5 and SHA-1 hashes for each message, measuring the time taken for computation, and verifying the integrity of the computed hashes. The performance analysis revealed the time taken for hash computation for different message sizes.