| **Course Name:** | **Applied Cryptography 116U01E628** | **Semester:** | **VI** |
| --- | --- | --- | --- |
| **Date of Performance:** | **09-04-2025** | **DIV/ Batch No:** | **C - 3** |
| **Student Name:** | **Romil Lodaya** | **Roll No:** | **16010122096** |

**Experiment No:9**

**Title:** Cryptographic Hash Functions and Applications using python libraries

| **Aim and Objective of the Experiment:** |
| --- |
| Cryptographic Hash Functions and Applications using python libraries |

| **COs to be achieved:** |
| --- |
| **CO3**  Comprehend cryptographic hash functions, Message Authentication Codes and Digital Signatures for Authentication |

| **Books/ Journals/ Websites referred:** |
| --- |
| Stallings, W., Cryptography and Network Security: Principles and Practice, Second edition, Person Education  Forouzan, B. A. (2018). Cryptography and Network Security. McGraw-Hill Education. |

| **Theory:** Explain the following. |
| --- |
| A hash function is a mathematical algorithm that converts data of any size into a fixed-size output, called a hash value or digest, which is unique to that input.  Purpose:  Hash functions are used for various purposes, including data integrity checks, digital signatures, password storage, and creating unique identifiers.  One-way function:  A good hash function is designed to be a one-way function, meaning it's easy to compute the hash value from the input, but computationally infeasible to reverse the process and determine the original input from the hash value.  Fixed-size output:  Regardless of the input size, the hash function always produces an output of a fixed size.  Collision resistance:  Ideally, a good hash function should minimize the chances of two different inputs producing the same hash value (a collision).  Examples:  Cryptographic hash functions: SHA-256, SHA-512, MD5.  Non-cryptographic hash functions: Used in hash tables and other data structures for efficient data storage and retrieval.  Applications:  Data Integrity: Verifying that data has not been tampered with by comparing the hash of the original data with the hash of the received data.  Digital Signatures: Creating a unique "fingerprint" of a document or message to ensure authenticity and non-repudiation.  Password Storage: Storing password hashes instead of plain text passwords to protect against unauthorized access.  Data Structures: Used in hash tables and other data structures to efficiently store and retrieve data.  Rabin-Karp Algorithm: A string matching algorithm that uses hashing to find patterns in text. |

| Code and Output : |
| --- |
| **CODE:**  import hashlib  *data* = input("Enter the text: ")  *md5\_hash* = hashlib.md5(*data*.encode()).hexdigest()  *sha256\_hash* = hashlib.sha256(*data*.encode()).hexdigest()  *sha512\_hash* = hashlib.sha512(*data*.encode()).hexdigest()  print("MD5:", *md5\_hash*)  print("SHA-256:", *sha256\_hash*)  print("SHA-512:", *sha512\_hash*)  **OUTPUT:** |

Post Lab Questions:

1. Differences in MD5 and SHA family.

| **Feature** | **MD5** | **SHA Family (e.g., SHA-1, SHA-256, SHA-512)** |
| --- | --- | --- |
| **Hash Length** | 128 bits (32 hex characters) | SHA-1: 160 bits  SHA-256: 256 bits  SHA-512: 512 bits |
| **Security** | Weaker, prone to collisions | More secure (especially SHA-256 and SHA-512) |
| **Speed** | Faster | Slower (due to higher complexity and larger output) |
| **Collision Resistance** | Poor (collisions found easily) | Stronger resistance (especially SHA-2 family) |
| **Usage** | Legacy systems | Current cryptographic standards (e.g., SSL/TLS, blockchain) |

| Conclusion: |
| --- |
| Cryptographic hash functions ensure data integrity, security, and authentication. SHA is stronger and more secure than MD5 for modern use. |