Cyber Security Fundamentals

CSE 866

Sparsh Salodkar

IMT2022113

Comparison Between Hashing Algorithms

(MD5, SHA-1, and SHA-256)

Introduction

Hashing is the process of converting input data into a fixed-length string of characters called a hash value. It acts as a unique fingerprint for a file. If a file changes, even by one bit, its hash changes drastically. This property allows hashing to detect file tampering and verify data integrity. In this project, we manually implemented three popular hashing algorithms — MD5, SHA-1, and SHA-256 — using Python (without any cryptographic libraries) and compared their results on multiple files.

Objective

- To understand how hashing ensures data integrity.
- To compare different hashing algorithms (MD5, SHA-1, SHA-256).
- To show how file tampering can be detected using hash comparison.

Tools Used

- **Language:** Python 3
- **Libraries:** Typer, Tabulate, TQDM (for CLI, progress bar, and table)
- **Platform:** Command-line interface (terminal)

  NOTE : Libraries have only been used for aesthetics and tabulation, the actual encryption process has been manually done by code.

Methodology

The experiment was carried out in the following steps:

1. Three hashing algorithms (MD5, SHA-1, SHA-256) were manually coded in Python.
2. Several test files of different types were selected.
3. Each file was hashed using all three algorithms, and results were compared.
4. One file was modified slightly (tampered) to test if the hashes change.
5. Results were displayed in a table format for comparison.

Results

Table 1A: MD5 Hashes for Different Files

File Name	MD5
1.txt	00ee948b723378101239812546f4b8f3
2.png	3df4eb0157fd3ec4fdb7d6672564d81f
3.pdf	5ea2afe6fecc5431ed07e69f16a5f7cf

Table 1B: SHA-1 Hashes for Different Files

File Name	SHA-1
1.txt	36abec9deec51e82ef00267918b5337d87c46462
2.png	386e3e03f0df39a4702ca9b6c28afce89f262e58
3.pdf	bdca498d4fead09ad70d3feb9eae9df4d835d8cb

Table 1C: SHA-256 Hashes for Different Files

File Name	SHA-256
1.txt	763c0875e8c974a66e66f9b367e3bd1e4c86e7d4bbec140d3414989b6ca5af48
2.png	f9f6fcec817f5b592302e2b16fffe07de8c99e9846caed9c6698a5b9495afd5b
3.pdf	0ea8318da58016e1b185bb68a9bab86c93eb2da0de1ca536fdf782a9a89e6089

Comparisons

Table 2A: MD5 Comparison for 2.png (Original vs Tampered)

File Name	MD5
2.png	3df4eb0157fd3ec4fdb7d6672564d81f
2_tampered.png	f428e58d39b4fd08aa5a620aa3b84f35

Table 2B: SHA-1 Comparison for 2.png (Original vs Tampered)

File Name	SHA-1
2.png	386e3e03f0df39a4702ca9b6c28afce89f262e58
2_tampered.png	b1462ed6a36513bccad4310eb5d55ba4463777a5

Table 2C: SHA-256 Comparison for 2.png (Original vs Tampered)

File Name	SHA-256
2.png	f9f6fcec817f5b592302e2b16fffe07de8c99e9846caed9c6698a5b9495afd5b
2_tampered.png	982ec26756149cfdf80018444b1f49501f2d1f8c723fc19dd1fa96a357c3d215

Discussion

The results show that the hash value of each file is unique. When a file was tampered with (a single-bit change), the resulting hash values were entirely different. This confirms the **avalanche effect** — a small input change causes a large, unpredictable change in the hash output. Among the algorithms tested:

- **MD5** – Fast but insecure due to collisions.
- **SHA-1** – More secure than MD5 but still weak by modern standards.
- **SHA-256** – Strongest among the three, with high resistance to collisions and used in real-world security systems.

Conclusion

Hashing provides an effective way to ensure file integrity and detect tampering. This project clearly shows that even a small modification in data changes all three hashes completely. While MD5 and SHA-1 are outdated due to collision vulnerabilities, SHA-256 remains secure and reliable for modern applications. Thus, SHA-256 is recommended for data integrity and verification tasks.