Task 7: Keyed Hash and HMAC - Lab Report

Objective

To understand the properties of one-way hash functions, particularly the avalanche effect, by analyzing MD5 and SHA256 hash functions.

Files Created

1. `textfile.txt` - Original text file
2. `textfile_modified.txt` - Modified text file (1 character changed)
3. `hash_analyzer.cpp` - Program to analyze bit differences between hash values

Step 1: Create Original Text File

Content of textfile.txt:

```
This is a sample text file for hash function analysis.
We will test the avalanche effect of MD5 and SHA256 hash functions.
The avalanche effect means that a small change in input should produce
a dramatically different output hash value.
This property is crucial for cryptographic hash functions.
Information security is the practice of protecting information
by mitigating information risks and vulnerabilities.
```

Step 2: Generate H1 (Original Hash Values)

Commands:

```
openssl dgst -md5 textfile.txt
openssl dgst -sha256 textfile.txt
```

Output:

```
MD5(textfile.txt)= d05a9d071a556f5179847c89e9a4d791
SHA2-256(textfile.txt)=
cc6ed5ca06bb85101dd3ad653a880d02980d6c1a820cccb037f6e9cea6005ff5
```

Step 3: Modify Input File (Flip One Bit)

Modification: Changed 'analysis' to 'analysys' (i → y)

- This represents a single character change that affects multiple bits

Modified content:

```
This is a sample text file for hash function analysys.
[rest of content remains the same]
```

Step 4: Generate H2 (Modified Hash Values)

Commands:

```
openssl dgst -md5 textfile_modified.txt
openssl dgst -sha256 textfile_modified.txt
```

Output:

```
MD5(textfile_modified.txt)= b65ef3f75111e0d6e701378a21ff38f0
SHA2-256(textfile_modified.txt)=
791581338b928ac8352e753d18ceb651a460453a25f4fd8890df531a1346eb63
```

Step 5: Hash Comparison Analysis

Hash Values Summary:

Algorithm	Original (H1)
MD5	d05a9d071a556f5179847c89e9a4d791
SHA256	cc6ed5ca06bb85101dd3ad653a880d02980d6c1a820cccb037f6e9cea6005ff5

Bit-Level Analysis Results:

MD5 Analysis:

- Total bits: 128
- Same bits: 67
- Different bits: 61
- Similarity percentage: 52.34%
- Difference percentage: 47.66%

SHA256 Analysis:

- Total bits: 256
- Same bits: 128
- Different bits: 128
- Similarity percentage: 50.00%
- Difference percentage: 50.00%

Observations

1. Avalanche Effect Demonstration

Both hash functions demonstrate excellent avalanche effect:

- **MD5**: 47.66% of bits changed (61 out of 128 bits)
- **SHA256**: 50.00% of bits changed (128 out of 256 bits)

2. Similarity Analysis

- The hash values appear completely different despite only one character change in input
- No obvious pattern or relationship between H1 and H2 is visible
- Both algorithms show near-ideal 50% bit change ratio

3. Cryptographic Properties

- **Deterministic**: Same input always produces same output
- **Avalanche Effect**: Small input changes cause large output changes
- **Uniformity**: Changed bits appear randomly distributed
- **Irreversible**: No way to determine original input from hash

Conclusion

The experiment successfully demonstrates the avalanche property of cryptographic hash functions:

1. **Strong Avalanche Effect**: Both MD5 and SHA256 show approximately 50% bit changes for a single character modification
2. **Unpredictable Changes**: The changed bits appear randomly distributed
3. **Security Implication**: This property makes it computationally infeasible to find collisions or reverse-engineer inputs

The SHA256 algorithm shows perfect 50% change, while MD5 shows 47.66% change, both indicating strong avalanche properties essential for cryptographic security.

Bonus Program

Created `hash_analyzer.cpp` that:

- Converts hexadecimal hash values to binary
- Counts identical bits between H1 and H2
- Calculates Hamming distance (different bits)
- Provides statistical analysis of the avalanche effect

Compilation and Execution:

```
g++ -o hash_analyzer hash_analyzer.cpp
./hash_analyzer
```