

# National Institute of Technology, Tiruchirappalli



**Department of Computer Applications**

## **Information Security Lab** ***Lab 6***

Submitted to:

Dr. Ghanshyam Bopche

Submitted by:

Akriti Upadhyay

205120007

MCA – 2<sup>nd</sup> Year

# ***Application of Hashing Algorithms***

## ***FILE VERIFICATION***

### **Using Hashing Algorithms for File verification:**

- File verification is the process of using an algorithm for verifying the integrity of a computer file, usually by checksum.
- Here, the approach is to generate a hash of the copied file and compare that to the hash of the original file. If the hashes match then the copied file is untampered, otherwise someone has tampered the file.

**Python Code:** fileVerification.ipynb

```
In [1]: ▶ # module for hashing algorithm implementation
import hashlib
```

#### Server Side

```
In [2]: ▶ # at the server end
# the untampered hashes of the original file ( actual values )
# calculated once for every version
origMD5 = "DDE2D6E30D0B6A87EDC921CDA5B462FB".lower()
origSHA1 = "F5CFCD595BB9BEA09BDCEB03CAFB8A80B30705D1".lower()
origSHA256 = "5C2E1745D0D899CFE948E02DB0BA8F2BBBD34147F6A24F119C514F389A4F3A04".lower()
```

```
In [3]: ▶ # function at server side to validate hashes
def validateHashes(calMD5, calSHA1, calSHA256):
    if( (origMD5 == calMD5) and (origSHA1 == calSHA1) and (origSHA256 == calSHA256) ) :
        return True
    return False
```

#### Common data between Client and Server

```
In [4]: ▶ # defining block size
# and functions for hash calculation ( can have multiple and different functions )
# common for both client and server side
blockSize = 2**20
md5 = hashlib.md5()
sha1 = hashlib.sha1()
sha256 = hashlib.sha256()
```

#### Client Side Application

```
In [26]: ▶ # function calculating hash on client side
def hashFile(inputFile):
    """
    Calculate MD5, SHA1 and SHA256 hash of the file.
    """
    with open(inputFile, 'rb') as clientCopy:
        while True:
            # read the file in block sizes
            dataBlock = clientCopy.read(blockSize)
            if not dataBlock:
                break
            md5.update(dataBlock)
            sha1.update(dataBlock)
            sha256.update(dataBlock)

    return md5.hexdigest().lower(), sha1.hexdigest().lower(), sha256.hexdigest().lower()
```

## Output for Untampered file:

```
In [27]: ▶ # get file name client side
inputFileName = input("Enter the filename: ")

Enter the filename: test.txt
```

```
In [28]: ▶ # function call to calculate hashes
rawMD5, rawSHA1, rawSHA256 = hashFile(inputFileName)
```

```
In [29]: ▶ if( validateHashes(rawMD5, rawSHA1, rawSHA256) ) :
print("Validation successful: Proceeding to startup...")
else :
    print("Application is tampered!! Exit the application!!")

Validation successful: Proceeding to startup...
```

## Output for Tampered file:

```
In [30]: ▶ # get file name client side
inputFileName = input("Enter the filename: ")

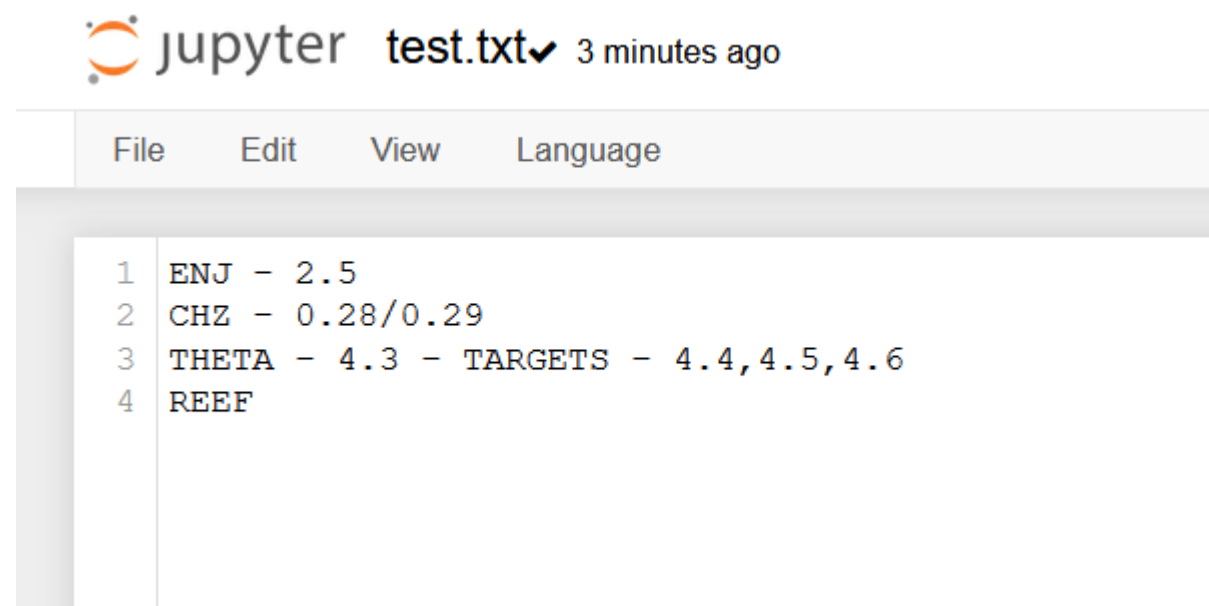
Enter the filename: tampered.txt

In [31]: ▶ # function call to calculate hashes
rawMD5, rawSHA1, rawSHA256 = hashFile(inputFileName)

In [32]: ▶ if( validateHashes(rawMD5, rawSHA1, rawSHA256) ) :
    print("Validation successful: Proceeding to startup...")
else :
    print("Application is tampered!! Exit the application!!")

Application is tampered!! Exit the application!!
```

## Contents of Untampered File:



jupyter test.txt ✓ 3 minutes ago

File	Edit	View	Language
1	ENJ - 2.5		
2	CHZ - 0.28/0.29		
3	THETA - 4.3 - TARGETS - 4.4,4.5,4.6		
4	REEF		

## Contents of Tampered File:

In this tampered file, the last 'targets' in line 3 has been changed from 4.6 to 4.7

File Edit View Language

```
1 ENJ - 2.5
2 CHZ - 0.28/0.29
3 THETA - 4.3 - TARGETS - 4.4,4.5,4.7
4 REEF
```

---

**X-X-X-X**