

**NAME** : Gogineni Hemanth Kumar

**REG NO** : 11909584

**ROLL NO** : 56

**COURSE CODE** : INT301

**COURSE TITLE** : OPEN SOURCE TECHNOLOGIES

**SECTION** : KE008

**TOPIC**  : Calculate MD5 and SHA1 and CRC32 (10

files) for making a comparison of files and

check all the files for integrity, internet

downloads and data tampering

**INTRODUCTION**

**OBJECTIVE**

Common techniques for determining a file's unique identification (or checksum) are MD5, SHA1, and CRC32. This identifier's objective is to make it possible for you to confirm that the file you have downloaded is the exact same one that was first uploaded to the internet. You may be sure that a file hasn't been tampered with during the download process if the identification of the file you downloaded matches the identifier of the original file. CRC32 is a cyclic redundancy check function, whereas MD5 and SHA1 are cryptographic hash functions. These methods function by generating a fixed-length hash value from the input file data. Because the hash value is specific to the file, even minor changes to the file's data will produce an entirely new hash value. Once you obtain the checksums for the original files, you can verify that they are identical by comparing them to the checksums of the files you downloaded. You can be sure that the files haven't been tampered with throughout the download process if the checksums match. You should try downloading the file again from a reliable source if the checksums do not match, as you may have obtained a corrupted or changed file.

**DESCRIPTION**

**MD5(MESSAGE DIGEST):** Creating a 128-bit message digest, this commonly used cryptographic hash function is utilised. In applications where data integrity is crucial, such as checksums and digital signatures, MD5 was created by Ronald Rivest in 1991. Now, though, it is thought to be susceptible to collision attacks, therefore more secure options like SHA-2 and SHA-3 are advised.

**SHA1(SECURE HASH ALGORITHM) :** Another well-liked cryptographic hashing algorithm that generates a 160-bit message digest is SHA1. It was created by the National Security Agency (NSA) of the United States and is frequently used in applications that demand data integrity, such as digital certificates. Yet it is also being phased out in favour of SHA-2 and SHA-3 because it is susceptible to collision attacks.

**CRC32(CYCLIC REDUNDANCY CHECK) :** Unlike MD5 and SHA1, CRC32 is a checksum technique used to identify data transmission mistakes rather than a cryptographic hash function. To validate the accuracy of data sent between computers, it generates a 32-bit checksum. CRC32 is frequently used in applications like ZIP files, Ethernet frames, and other data transmission protocols even though it is not as secure as cryptographic hash algorithms like MD5 and SHA1.

Any of these hash algorithms can be used to verify the authenticity of files downloaded over the internet or to spot data manipulation. Then, use one of the hash functions to create the hash value for the original file (MD5, SHA1, or CRC32). Compare this hash value to the hash of the downloaded file after that. The file is probably intact and wasn't altered during transmission or download if the hash values match. The file may have been altered or corrupted during transmission or download if the hash values do not match.

**SCOPE OF THE PROJECT**

**FILE COMPARISON :** In order to establish whether two or more files are identical, the project should be able to compare them. This could entail computing the hash values of the files using MD5, SHA1, or CRC32 and comparing them.

**INTEGRITY CHECK :** The project ought to be able to examine a file's integrity and detect whether it has been altered or tampered with. This may entail computing the file's MD5, SHA1, or CRC32 hash value and contrasting it with the original hash value.

**INTERNET DOWNLOADS :** Verifying the integrity of files acquired via the internet should be possible for the project. This may entail computing the hash value of the downloaded file using MD5, SHA1, or CRC32 and contrasting it with the hash value originally supplied by the source.

**DATA TAMPERING :** If any data has been altered or tampered with, the project should be able to identify it. Checking to determine if the hash values of the original and updated files match could be one way to do this.

**SYSTEM DESCRIPTION**

**Target system description**

**Hashing Algorithm :** Hashes are the output of a hashing algorithm like MD5 (Message Digest 5) or SHA (Secure Hash Algorithm). These algorithms essentially aim to produce a unique, fixed-length string – the hash value, or “message digest” – for any given piece of data or “message”. As every file on a computer is, ultimately, just data that can be represented in binary form, a hashing algorithm can take that data and run a complex calculation on it and output a fixed-length string as the result of the calculation. The result is the file’s hash value or message digest.

**MD5:** The MD5 (message-digest algorithm) hashing algorithm is a one-way cryptographic function that accepts a message of any length as input and returns as output a fixed-length digest value to be used for authenticating the original message.

The MD5 hash function was originally designed for use as a secure cryptographic hash algorithm for authenticating [digital signatures](https://www.techtarget.com/searchsecurity/definition/digital-signature). But MD5 has been deprecated for uses other than as a noncryptographic checksum to verify [data integrity](https://www.techtarget.com/searchdatacenter/definition/integrity) and detect unintentional data corruption. Although originally designed as a cryptographic [message authentication code](https://www.techtarget.com/searchsecurity/definition/message-authentication-code-MAC) algorithm for use on the internet, MD5 hashing is no longer considered reliable for use as a [cryptographic checksum](https://www.techtarget.com/searchsecurity/definition/cryptographic-checksum) because security experts have demonstrated techniques capable of easily producing MD5 collisions on commercial off-the-shelf computers. An encryption collision means two files have the same hash. Hash functions are used for message security, password security, computer forensics and cryptocurrency.

### **Message-digest algorithm characteristics**

Message digests, also known as hash functions, are one-way functions; they accept a message of any size as input and produce as output a fixed-length message digest.

MD5 is the third message-digest algorithm Rivest created. MD2, MD4 and MD5 have similar structures, but MD2 was optimized for 8-bit machines, in comparison with the two later algorithms, which are designed for 32-bit machines. The MD5 algorithm is an extension of MD4, which the critical review found to be fast but potentially insecure. In comparison, MD5 is not quite as fast as the MD4 algorithm, but offered much more assurance of data security.

### **How does MD5 work?**

The MD5 message-digest hashing algorithm processes data in 512-bit strings, broken down into 16 words composed of 32 bits each. The output from MD5 is a 128-bit message-digest value.

Computation of the MD5 digest value is performed in separate stages that process each 512-bit block of data along with the value computed in the preceding stage. The first stage begins with the message-digest values initialized using consecutive [hexadecimal](https://www.techtarget.com/whatis/definition/hexadecimal) numerical values. Each stage includes four message-digest passes, which manipulate values in the current data block and values processed from the previous block. The final value computed from the last block becomes the MD5 digest for that block.

### **Alternatives to MD5**

A major concern with MD5 is the potential it has for message collisions when message hash codes are inadvertently duplicated. MD5 hash code strings also are limited to 128 bits. This makes them easier to breach than other hash code algorithms that followed.

Alternate hash codes to MD5 include the following.

**1.Secure Hash Algorithm 1 (SHA-1**

**2.The SHA-2 hash code family**

**3.Cyclic redundancy check (CRC) codes**

**HOW TO FIND MD5 VALUE FOR ANY FILE**

Open a powershell window and try the following command:

Get-FileHash {filename} -Algorithm MD5

**SHA1:**

SHA-1 (short for Secure Hash Algorithm 1) is one of several cryptographic hash function .

It's most often used to verify a file has been unaltered. This is done by producing a checksum before the file has been transmitted, and then again once it reaches its destination. The transmitted file can be considered genuine only if both checksums are identical.

One real-world example where SHA-1 may be used is when you're entering your password into a website's login page. Although it happens in the background without your knowledge, it may be the method a website uses to securely verify that your password is authentic. If the website uses the SHA-1 cryptographic hash function, it means your password is turned into a checksum after you enter it in. That checksum is then compared with the checksum that's stored on the website that relates to your current password, whether you haven't changed your password since you signed up or if you just changed it moments ago. If the two match, you're granted access; if they don't, you're told the password is incorrect.

Another example where this hash function may be used is for file verification. Some websites will provide the checksum of the file on the download page so that when you download it, you can check the checksum for yourself to ensure that the downloaded file is the same as the one you intended to download.

**HOW TO FIND MD5 VALUE FOR ANY FILE**

Open a powershell window and try the following command:

Get-FileHash {filename} -Algorithm SHA1

**CRC32:**

**CYCLIC REDUNDANCY CHECK(CRC32):** is a data corruption detection error-detecting code. A brief checksum is computed based on the content of the data and delivered alongside it when it is sent. Checksums are generated once more and compared to sent checksums when data is received. The two must be equivalent in order for there to be no data corruption. A variable-length string is changed into an 8-character string by the CRC-32 algorithm itself.

A checksum algorithm called CRC-32 (Cyclic Redundancy Check-32) is frequently used to find mistakes in data transfer. A 32-bit checksum value is produced, which can be used to confirm the accuracy of the data.

The input data is divided into a number of "chunks" for the CRC-32 algorithm, and each chunk is subjected to a number of bitwise XOR and shifting operations. A 32-bit checksum value is added to the end of the data as the final output.In computer networks, storage devices, and other applications where data integrity is important, the CRC-32 method is frequently employed. It is a quick and trustworthy method that can find most data transfer issues. The checksum is produced using a variety of CRC-32 algorithm variants, each of which uses a different polynomial value. CRC-32/ISO-HDLC, CRC-32C, and CRC-32K are some of the more popular CRC-32 polynomials.

**HOW TO FIND MD5 VALUE FOR ANY FILE**

Open a powershell window and try the following command:

Get-FileHash {filename} -Algorithm CRC32

**ANALYSIS REPORT**

It is essential to **verify the authenticity of a file** to avoid incidences of downloading suspicious data and infecting your computer with ransomware, viruses, malware and other corrupt and harmful malicious components. One compromised file, even of a few bytes, can exploit the company’s entire IT infrastructure if it is not being detected and eradicated altogether.

## **Checking Integrity of a file**

Checking the integrity or hash value of a file is aims at checking that either a file is genuine or verifying that a file has not been modified by untrusted parties. **Checking file integrity** is vital for both individuals and organizations.Every single file on the internet is unique in its way and when it is changed or altered, either its hash value or its **digital signature change**. Therefore, analysts can use both digital signatures and hash values to check the integrity of a file.

One of the most effective ways you can determine whether or not a file has been changed in any way is to check its digital signature if it has one. However, if the digital signature method is out of the question, you can also check to see if a file is in its original state by looking at its hash value and comparing it against its original’s hash value. Every file is unique in its own way, and when you apply an algorithm known as a “cryptographic hash function” to it, a string of characters is returned to you – this string of characters is a hash value. A specific hash value is only valid for a specific file in a specific state. If a file is changed by even one byte and an algorithm is once again applied to it, the hash value that will be returned will be completely different.

MD5 or SHA family hashes are set of random strings that allow security professionals to make sure that the downloading file is not tampered with or not corrupted. Some operating systems such as Linux, MacOS, and Windows allow verification of downloaded files through their hashes. For example, Windows OS offers Windows PowerShell to do this.

In fact, every file has a unique data. The analysts apply a **Cryptographic Hash Function** to check the hash value of a program. This function verifies whether the file is in the current state or it has been modified by malicious actors. **Cryptographic Hash Function algorithm** works by comparing the file’s original and current hash values. And if a byte or even a piece of the file’s data has been changed, the original and current hash values will be different, and therefore you will know whether it’s the same file or not.

**CHECKING HASH VALUES FOR DOWNLOADED FILES**

**First file:**

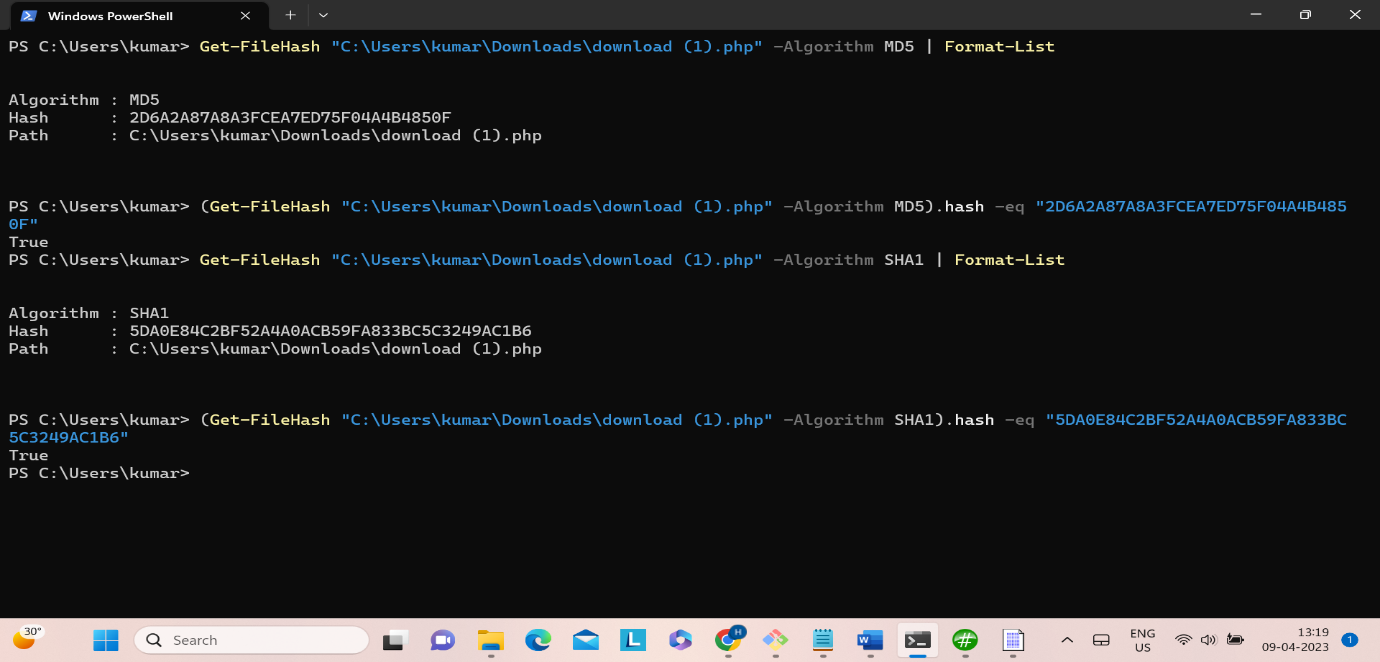
Text

Description automatically generated

File is not tampred

The hash values are same before and after download

**Second file:**



File is not tampred

The hash values are same before and after download

**Third file:**

Text

Description automatically generated

File is not tampred

The hash values are same before and after download

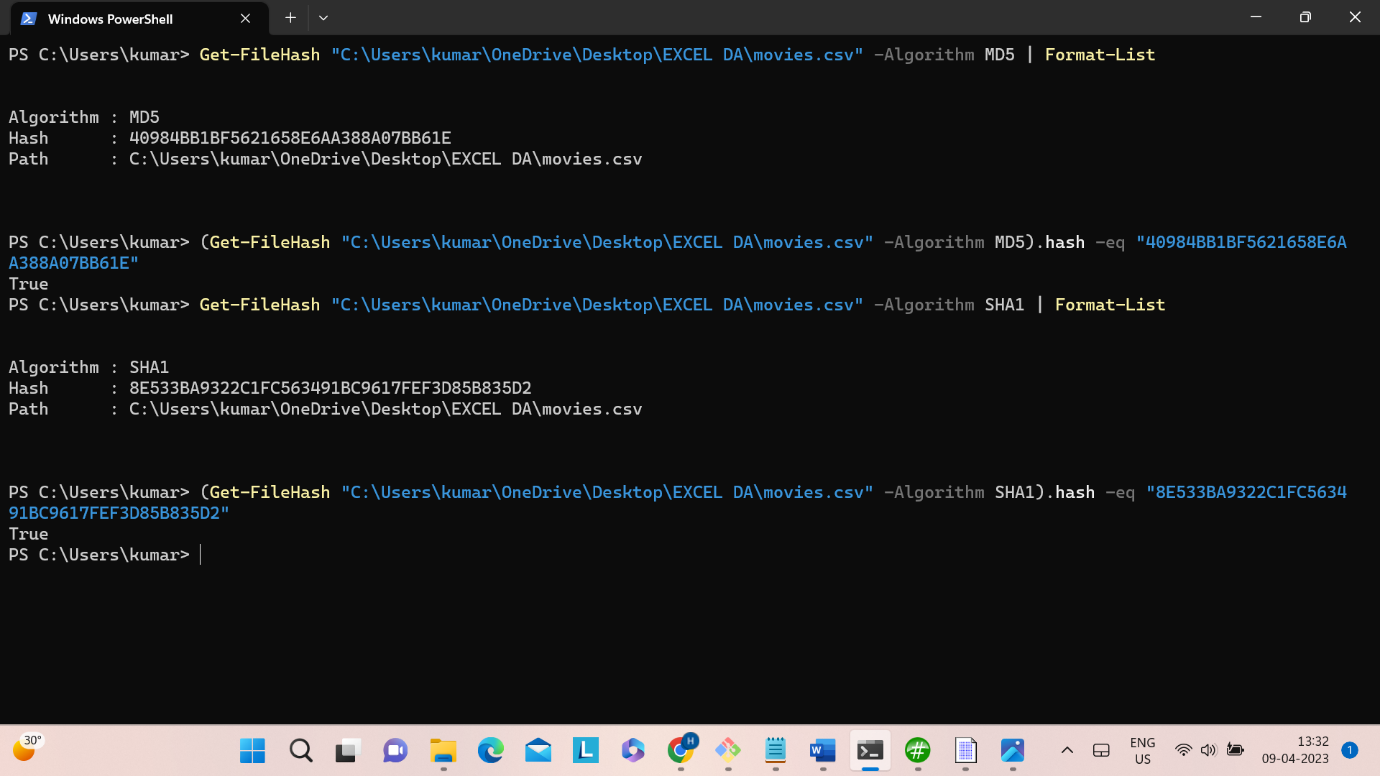
**Fourth file:**

Text

Description automatically generated

**Data not tampered, hash values are same before and after download**

**Fifth file:**

****

File is not tampred

The hash values are same before and after download

**Uing HashMyFiles Tool:**

Text, application

Description automatically generated

HashMyFiles is small utility that allows you to calculate the MD5 and SHA1 hashes of one or more files in your system. You can easily copy the MD5/SHA1 hashes list into the clipboard, or save them into text/html/xmlfile. HashMyFiles can also be launched from the context menu of Windows Explorer, and display the MD5/SHA1 hashes of the selected file or folder. By using these hash algorithms we can find out data is tampred or not.

**CONCLUSION**

As a result, it has been realized that checking file integrity is crucial to thwarting the penetration of cyber pests through a porous hole provided by malicious files. Therefore, it is vital to check the integrity of the file before downloading it to your PC, laptop, or any other device. The unauthentic file can result in the spread of ransomeware, Malware, Spyware, and other viruses. Any of the aforementioned tools can be used to generate a file's hash, which can then be compared to the source's hash or the original hash you generated previously to determine the file's integrity. The file has likely not been altered and is likely to be intact if the two hashes match.

**REFERENCES**

1. The MD5 Message-Digest Algorithm : RFC 1321
2. NIST Special Publication 800-131A Transitions: Guideline for Transitioning the Usage of Cryptographic Algorithms and Key Lengths
3. FIPS 180-4: Secure hash standard
4. RFC 3174: Use secure hash algorithm 1 (SHA1)
5. 8802-3:1996 ISO/IEC Information technology — Local and wide-area networks — Telecommunications and information sharing between systems Part 3: Physical layer specifications and the carrier sense multiple access with collision detection (CSMA/CD) access method