Project Report

on

[OPEN-SOURCE TECHNOLOGY]

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

For

[INT-301]

Submitted By

[ASTIK JAIN]

[11911098]

Submitted to

Rajeshwar Sharma

Assistant Professor

LOVELY FACULTY OF TECHNOLOGY & SCIENCES

LOVELY PROFESSIONAL UNIVERSITY

PUNJAB

April 2023

INTRODUCTION

MD5 :- MD5 (Message Digest 5) is a cryptographic hash function that produces a fixed-size output of 128 bits. It is widely used in computer security applications to verify the integrity of files, messages, and data transfers.

MD5 works by taking an input (a file or message) of any length and generating a unique fixed-size output that represents a "fingerprint" of the input data. This output is often referred to as the MD5 checksum or hash value. The checksum is unique to the input data, meaning that even a small change to the input data will result in a different MD5 checksum.

MD5 is commonly used to verify that a file has not been tampered with during transmission or storage. The MD5 checksum of the original file can be compared with the MD5 checksum of the received file to ensure that they are identical. If the checksums do not match, it indicates that the file has been corrupted or modified during transmission, and the file should be discarded or resent.

However, it is important to note that MD5 is no longer considered secure for cryptographic purposes due to various vulnerabilities that have been discovered. For this reason, more secure hash functions such as SHA-256 and SHA-3 are recommended for cryptographic applications.

In addition to verifying the integrity of files and messages, MD5 has been used for other applications such as password storage and digital signatures. In password storage, MD5 is used to generate a hash of the user's password, which is stored in a database instead of the actual password. When a user enters their password, the entered password is hashed and compared with the stored hash value to verify their identity.

MD5 can also be used for digital signatures, which are used to verify the authenticity and integrity of electronic documents. In this case, the MD5 hash value of a document is encrypted using the sender's private key to create a digital signature. The recipient can then use the sender's public key to decrypt the signature and compare it to the MD5 hash value of the received document to verify its authenticity and integrity.

Despite its widespread use in various applications, MD5 has been found to have several vulnerabilities that make it susceptible to attacks. One of the most significant vulnerabilities is that it is susceptible to collisions, which occur when different input data produces the same MD5 hash value. This can be exploited by attackers to create malicious files or messages with the same MD5 hash value as legitimate files or messages.

For this reason, more secure hash functions such as SHA-256 and SHA-3 are recommended for cryptographic applications. These hash functions have larger output sizes and are less susceptible to collisions, making them more secure for cryptographic purposes. However, MD5 is still commonly used for non-cryptographic purposes such as file verification and password storage.

SHA-1 :- SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function that produces a fixed-size output of 160 bits. It is widely used in computer security applications to verify the integrity of files, messages, and data transfers.

SHA-1 works by taking an input (a file or message) of any length and generating a unique fixed-size output that represents a "fingerprint" of the input data. This output is often referred to as the SHA-1 checksum or hash value. The checksum is unique to the input data, meaning that even a small change to the input data will result in a different SHA-1 checksum.

SHA-1 is commonly used to verify that a file has not been tampered with during transmission or storage. The SHA-1 checksum of the original file can be compared with the SHA-1 checksum of the received file to ensure that they are identical. If the checksums do not match, it indicates that the file has been corrupted or modified during transmission, and the file should be discarded or resent.

SHA-1 is also used for digital signatures, which are used to verify the authenticity and integrity of electronic documents. In this case, the SHA-1 hash value of a document is encrypted using the sender's private key to create a digital signature.

The recipient can then use the sender's public key to decrypt the signature and compare it to the SHA-1 hash value of the received document to verify its authenticity and integrity.

However, like MD5, SHA-1 has been found to have several vulnerabilities that make it susceptible to attacks. One of the most significant vulnerabilities is that it is susceptible to collisions, which occur when different input data produces the same SHA-1 hash value. This can be exploited by attackers to create malicious files or messages with the same SHA-1 hash value as legitimate files or messages.

For this reason, more secure hash functions such as SHA-256 and SHA-3 are recommended for cryptographic applications. These hash functions have larger output sizes and are less susceptible to collisions, making them more secure for cryptographic purposes. Despite this, SHA-1 is still commonly used for non-cryptographic purposes such as file verification and digital signatures.

Although SHA-1 is no longer considered secure for cryptographic purposes, it is still widely used in many applications. For example, it is used in many internet protocols such as TLS/SSL, SSH, and IPsec to provide secure communication between servers and clients. It is also used in many operating systems and software applications to authenticate software updates and verify the integrity of downloaded files.

In recent years, there have been several high-profile attacks on SHA-1, including the first successful collision attack in 2017. As a result, many organizations and software developers have transitioned to more secure hash functions such as SHA-256 and SHA-3. However, SHA-1 is still in use in many legacy systems and applications, and it may take time for all systems to transition to newer and more secure hash functions.

Overall, while SHA-1 is no longer considered secure for cryptographic purposes, it still has many uses in non-cryptographic applications such as file verification and software updates. As with any cryptographic tool, it is important to use SHA-1 and other hash functions properly and to stay up to date with the latest security recommendations and best practices to ensure the security and integrity of data and communications.

<u>CRC32</u> :- CRC32 (Cyclic Redundancy Check 32) is a type of error-detecting code that is commonly used in digital communication and storage systems. It is used to detect errors in data transmissions and to ensure the integrity of data stored on disks and other storage media.

CRC32 works by generating a fixed-size checksum of 32 bits based on the data being transmitted or stored. This checksum is calculated using a polynomial function that divides the data by a predetermined divisor. The resulting remainder is used as the checksum for the data. When the data is received or read back from storage, the same polynomial function is applied to the data, and the resulting remainder is compared with the original checksum to check for errors.

CRC32 is widely used in computer networking protocols such as Ethernet, TCP/IP, and Wi-Fi to ensure the integrity of data transmissions. It is also used in storage devices such as hard drives, USB drives, and memory cards to detect and correct errors that may occur during the reading or writing of data.

One advantage of CRC32 is that it is relatively fast and efficient compared to other error detection codes. It can be implemented in hardware or software and can be easily calculated using simple mathematical operations. However, CRC32 is not as secure as cryptographic hash functions such as MD5 and SHA-1 and should not be used for cryptographic purposes.

In summary, CRC32 is a type of error-detecting code that is widely used in digital communication and storage systems to ensure the integrity of data transmissions and stored data. While not suitable for cryptographic applications, it is fast, efficient, and widely supported in many computer systems and networking protocols.

One of the key advantages of CRC32 is its simplicity and efficiency. It can be implemented using relatively simple hardware or software and can be calculated quickly using simple mathematical operations. This makes it an ideal choice for applications where speed and efficiency are important.

Another advantage of CRC32 is its ability to detect a wide range of errors. While it cannot correct errors, it can detect errors caused by a wide range of factors such as noise, interference, or data corruption. This makes it an important tool for ensuring data integrity in a wide range of applications.

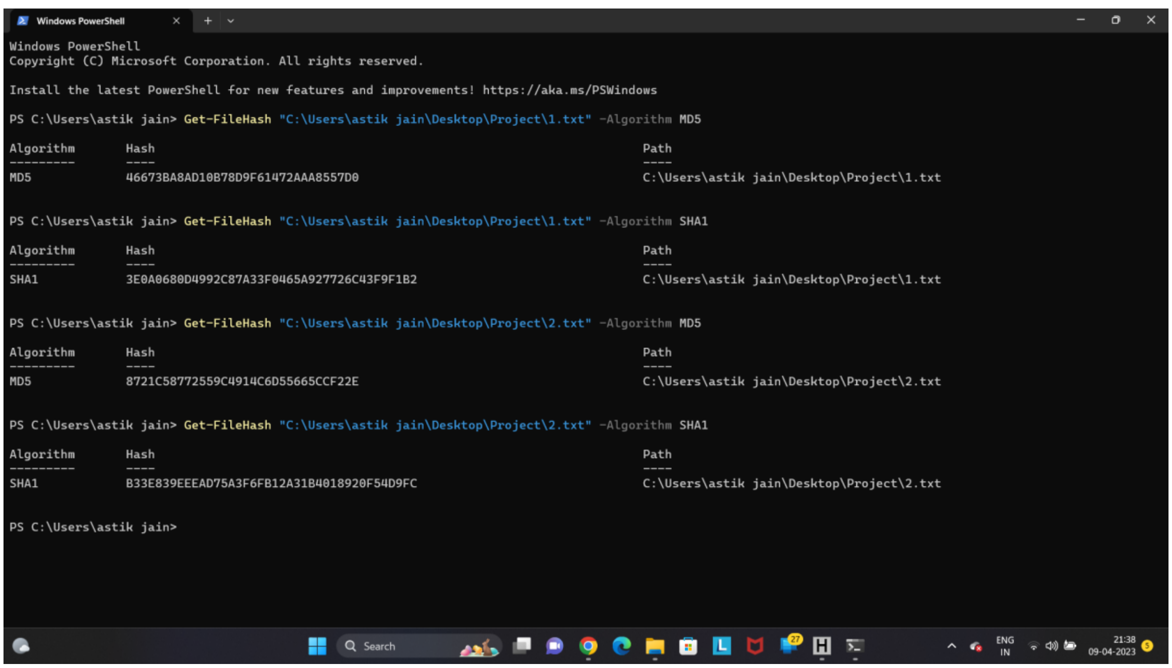
However, there are some limitations to the use of CRC32. One major limitation is that it is not a cryptographic hash function and does not provide any protection against intentional tampering or attacks. It can only detect errors caused by unintentional data corruption.

In addition, CRC32 is vulnerable to certain types of attacks, particularly those that involve intentionally creating collisions. This means that attackers may be able to create two different sets of data that have the same CRC32 checksum. For this reason, CRC32 should not be used for cryptographic purposes or applications where intentional tampering is a concern.

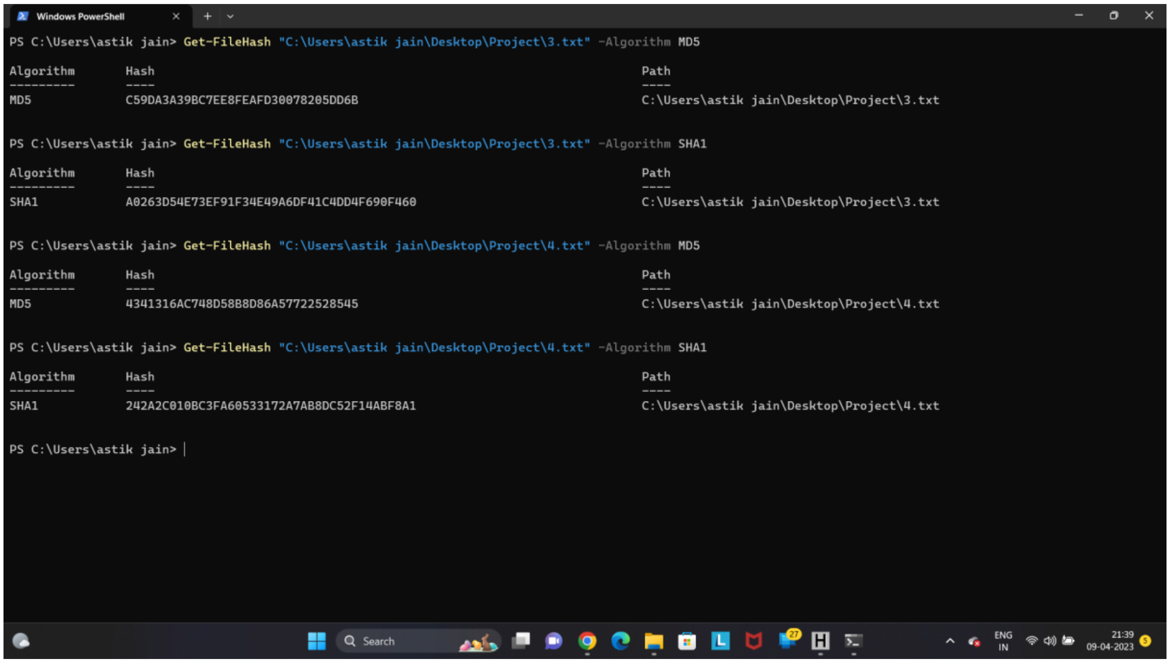
Overall, CRC32 is a widely used error-detecting code that provides an efficient and effective means of detecting errors in data transmissions and storage. While it is not suitable for cryptographic applications, it remains an important tool for ensuring the integrity of data in many different contexts.

ANALYSIS REPORT

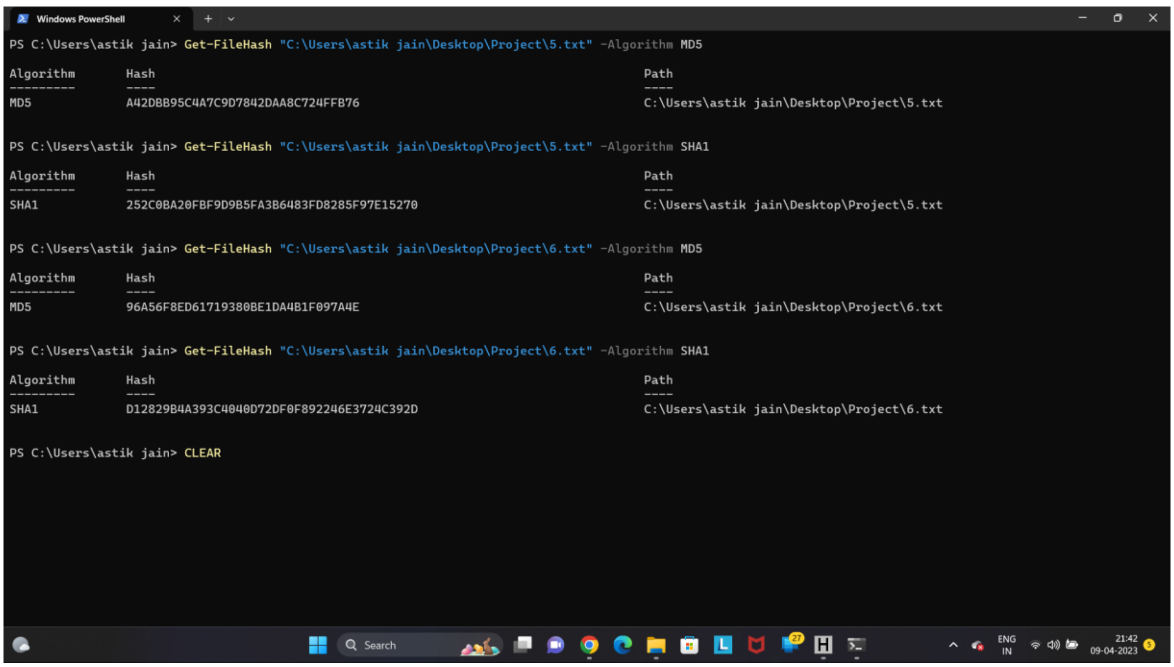
File 1 & 2 :- This is the MD5 and SHA1 for exisiting file.



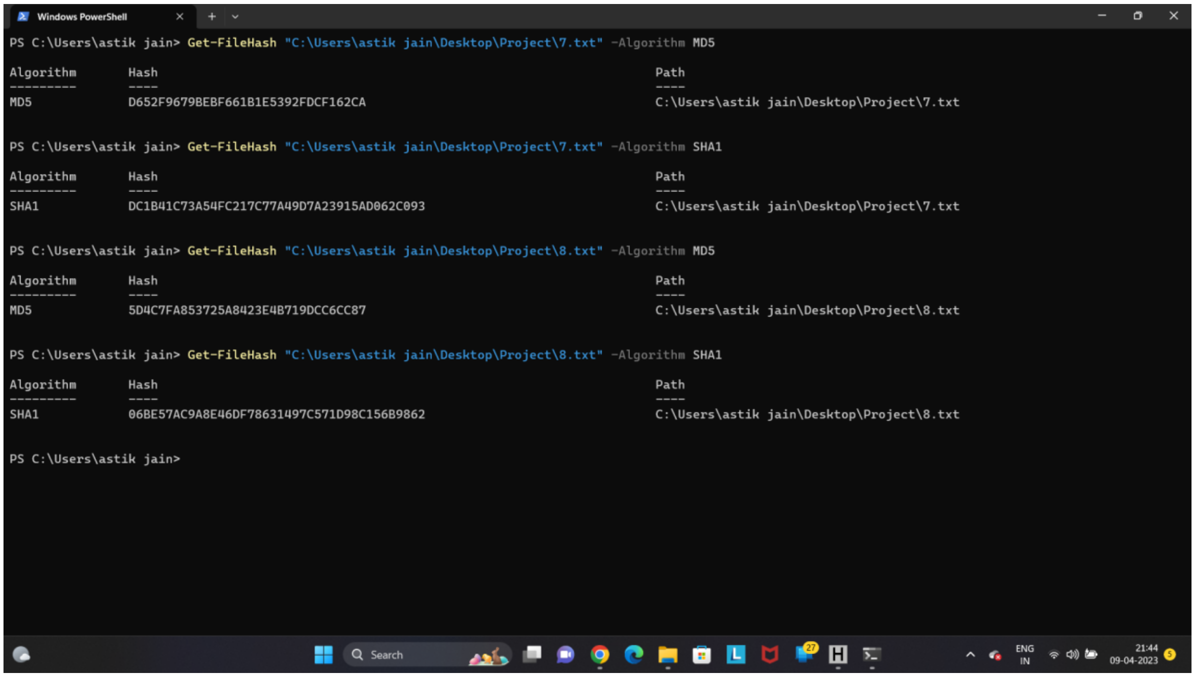
File 3 & 4 :- This is the MD5 and SHA1 for exisiting file.



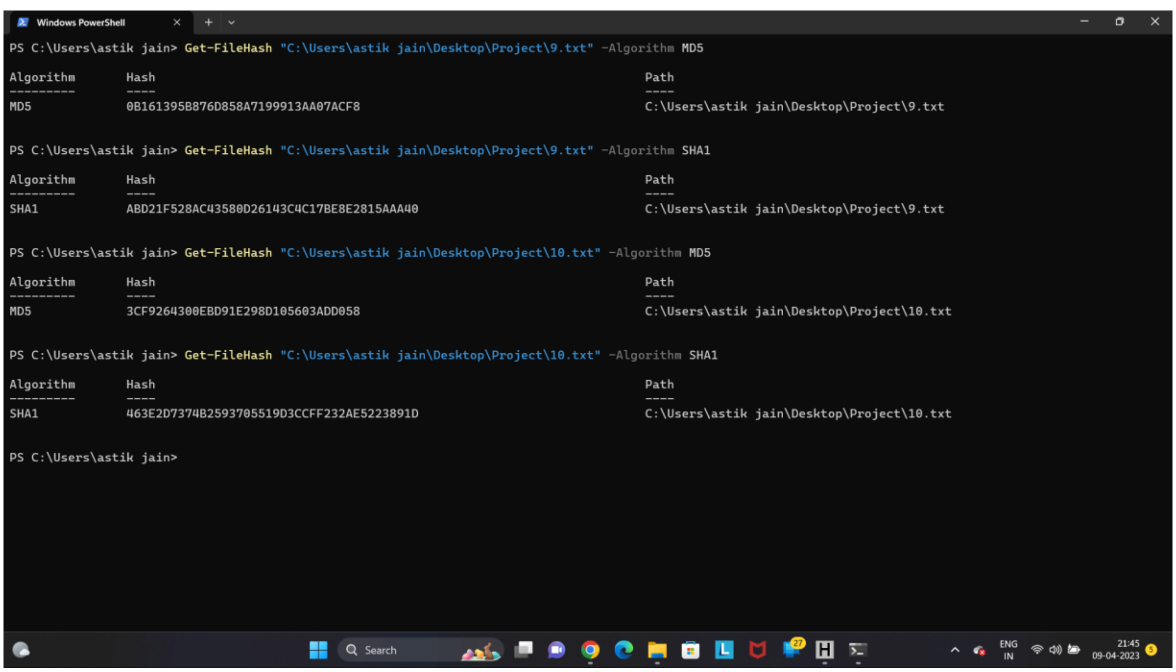
File 5 & 6 :- This is the MD5 and SHA1 for exisiting file.



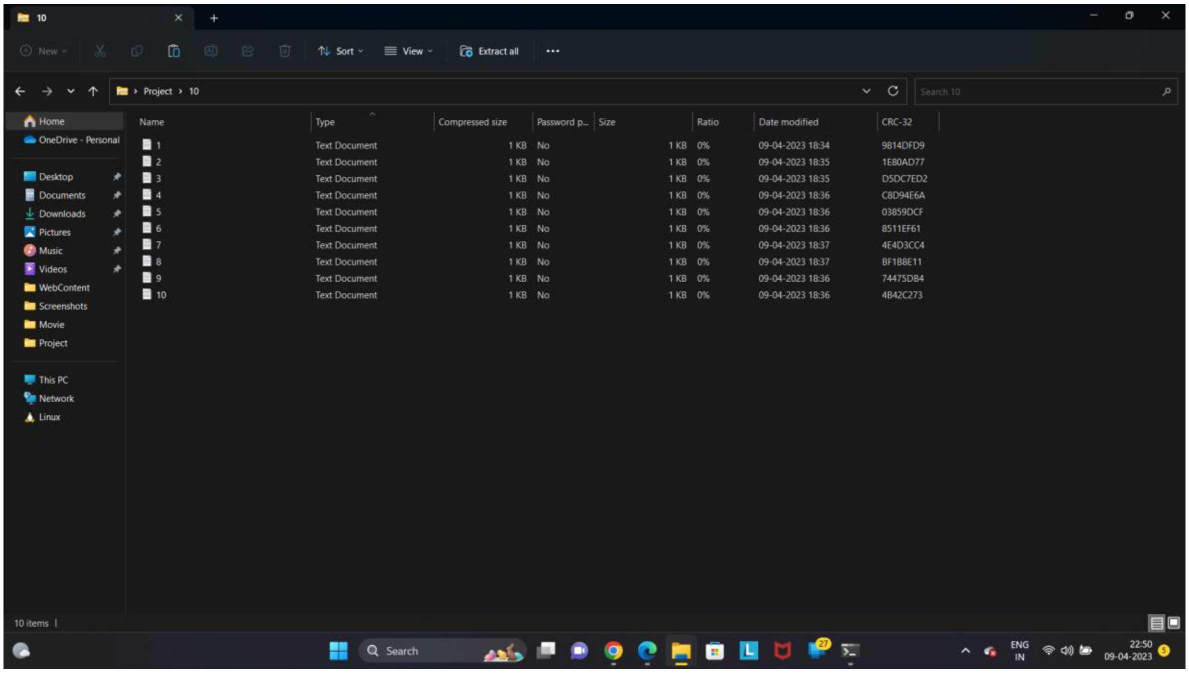
File 7 & 8 :- This is the MD5 and SHA1 for exisiting file.



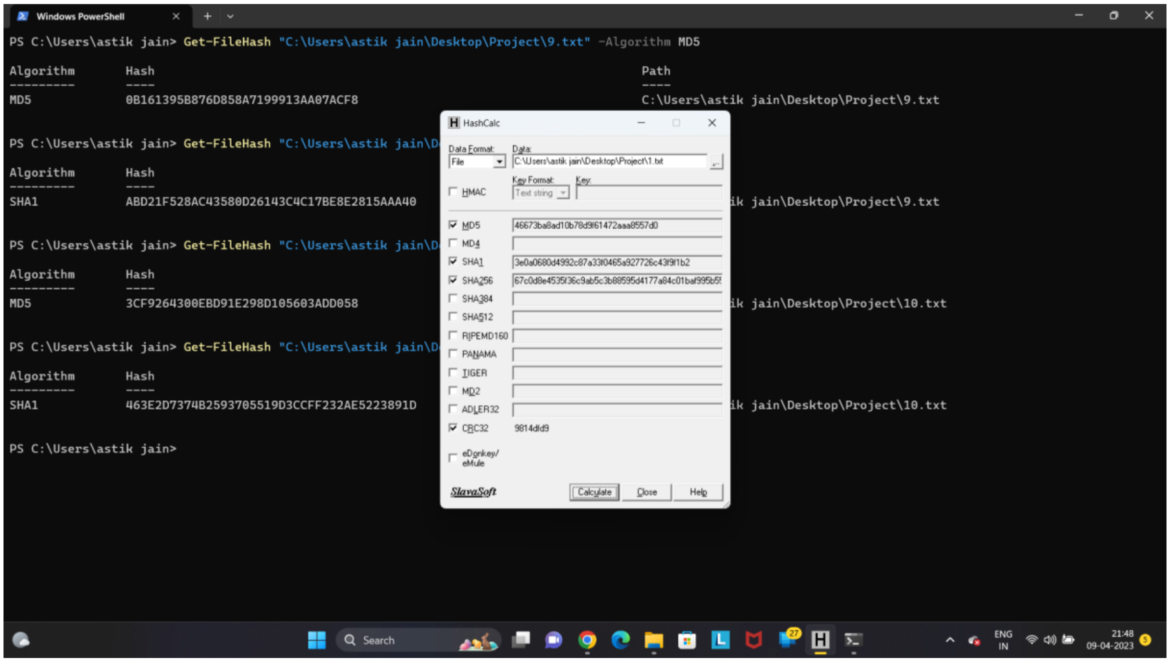
<u>File 9 & 10</u> :- This is the MD5 and SHA1 for exisiting file.

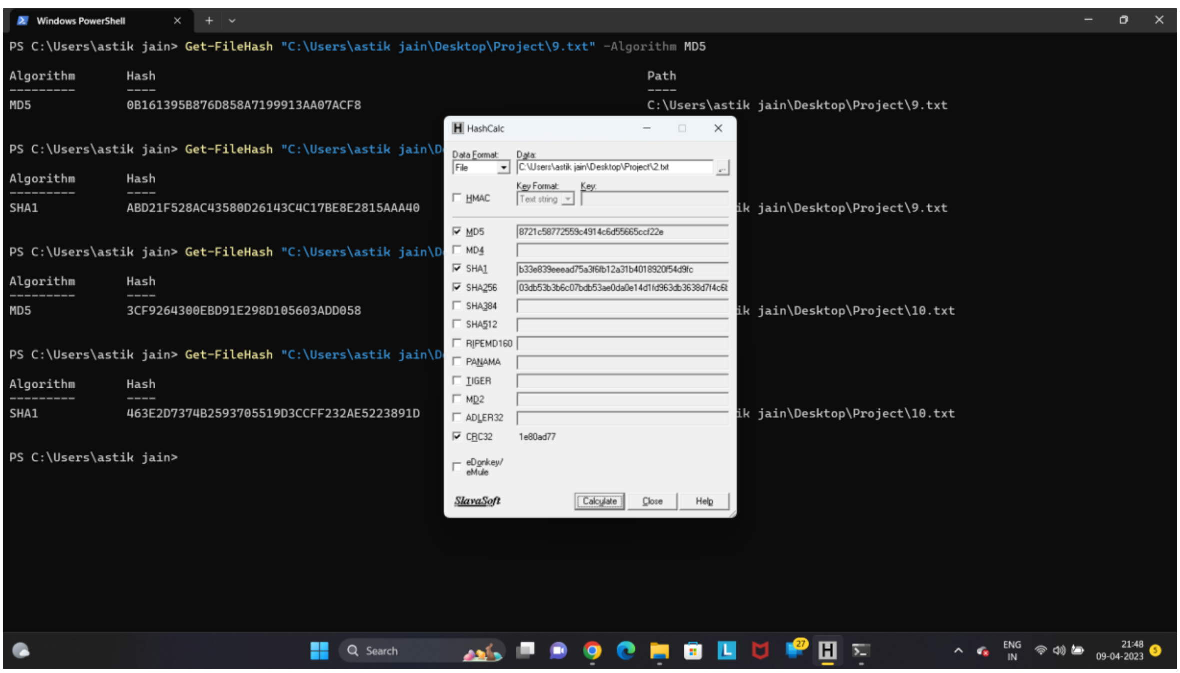


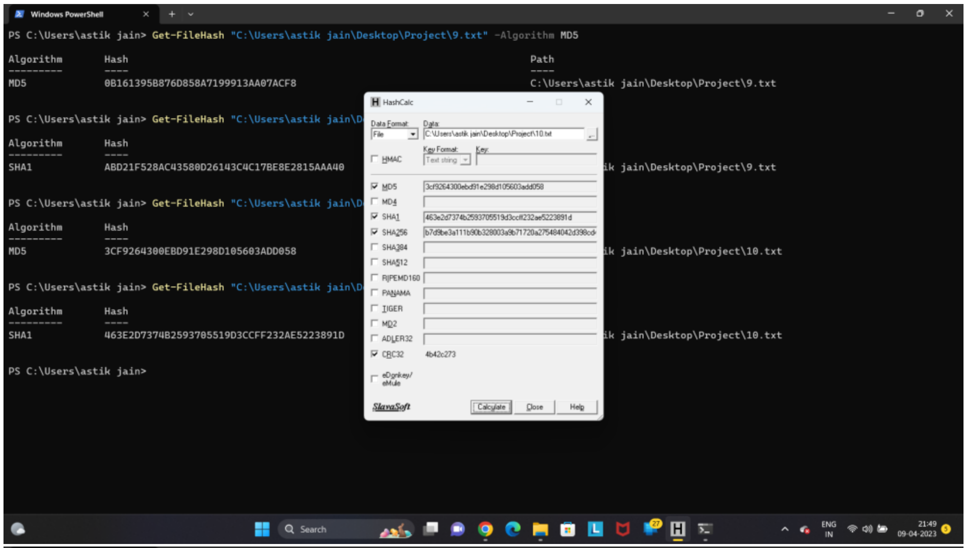
<u>CR32</u> :- These are CR32 through the source files using windows explorer.-

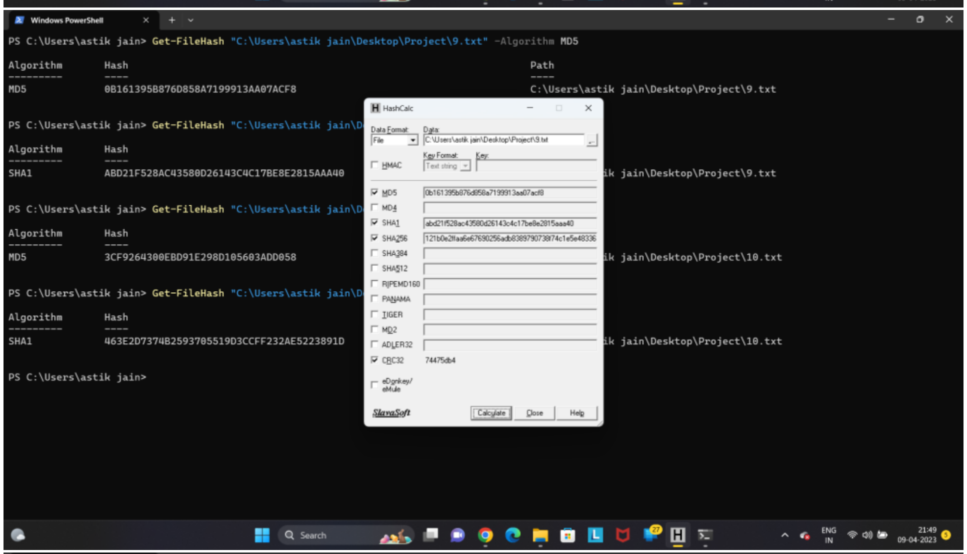


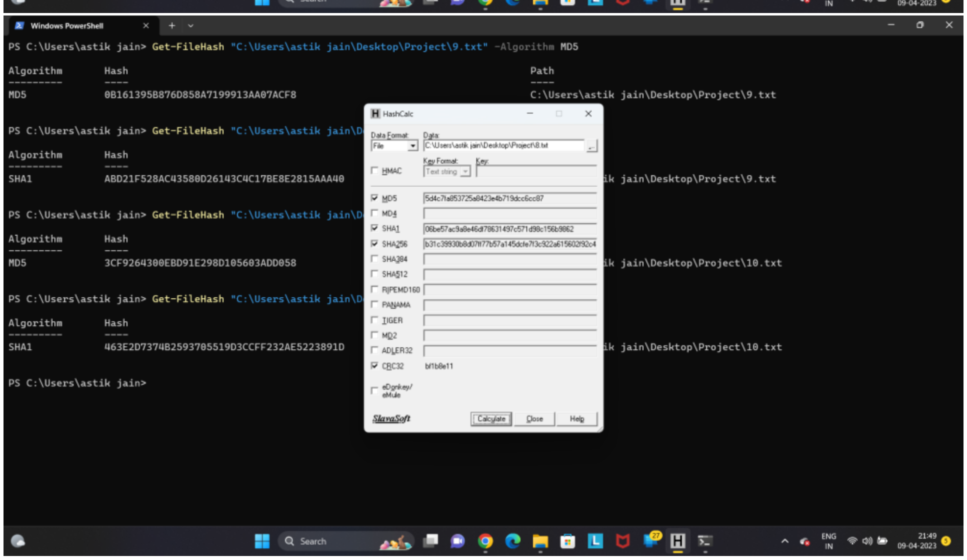
<u>HASH-CALCULATOR: -</u> In this process I have downloaded an open software in which I have just insert the file and through calculator I have just received the values and through powershell and windows operating system I have verified the value from original source and through open-source software. This is the proof for that.

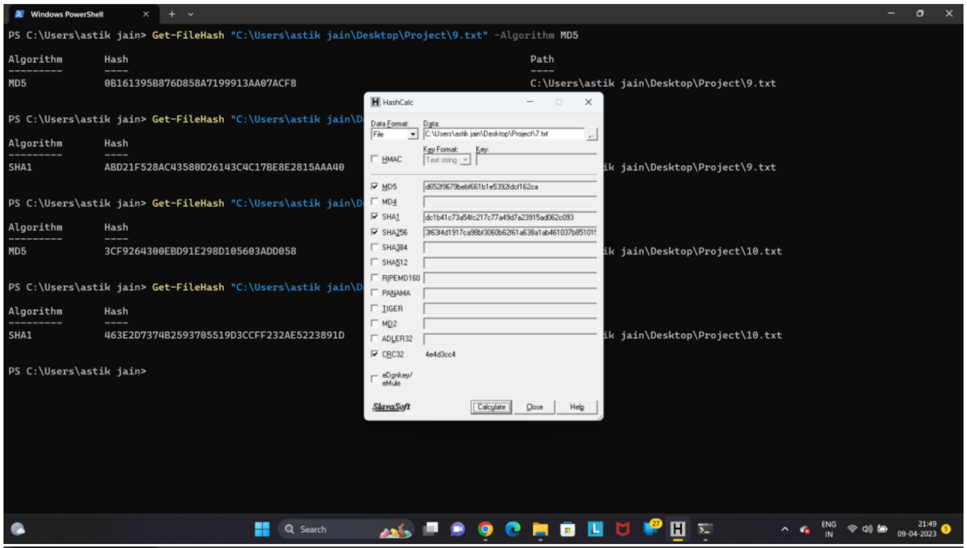
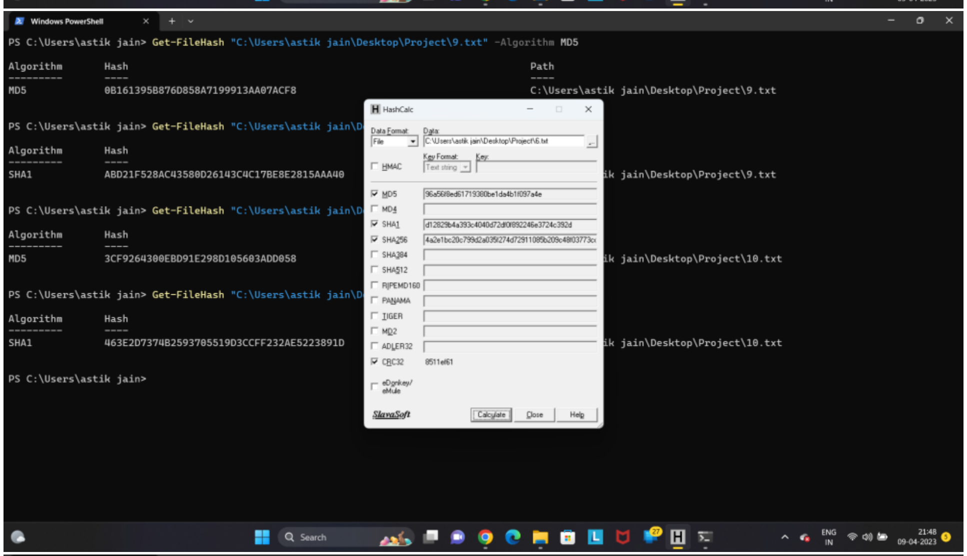
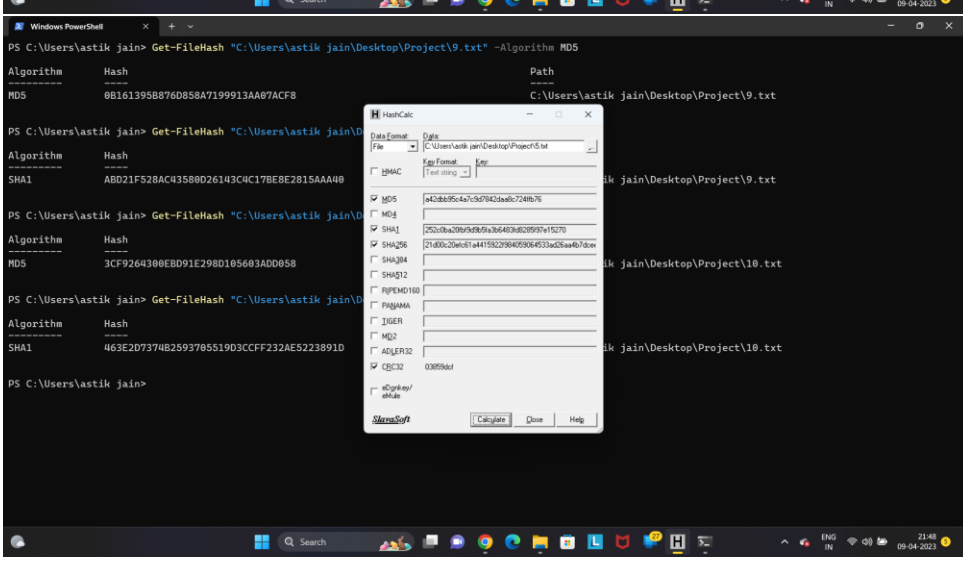

Windows PowerShell
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\Desktop\Project\9.txt" -Algorithm MD5
Algorithm Hash Path
MD5 0B161395B876D858A7199913AA07ACF8 C:\Users\astik jain\Desktop\Project\9.txt
SHA1 ABD21F528AC43580D26143C4C17BE8E2815AAA40
MD5 3CF9264300EBD91E298D105603ADD058
SHA1 463E2D7374B2593705519D3CCFF232AE5223891D
PS C:\Users\astik jain>
HashCalc
Data Format: File
Data: C:\Users\astik jain\Desktop\Project\1.txt
HMAC
Key Format: Text string
Key:
MD5 46673ba8ad10b78d9f61472aaa8557d0
MD4
SHA1 3e0a0680d4992c87a33f0465a927726c43f9f1b2
SHA256 67c0d8e4535f36c9ab5c3b88595d4177a84c01baf995b5
SHA384
SHA512
RIPEMD160
PANAMA
TIGER
MD2
ADLER32
CRC32 9814dfd9
eDonkey/ eMule
SlavaSoft
Calculate
Close
Help
Search
ENG IN
21:48 09-04-2023

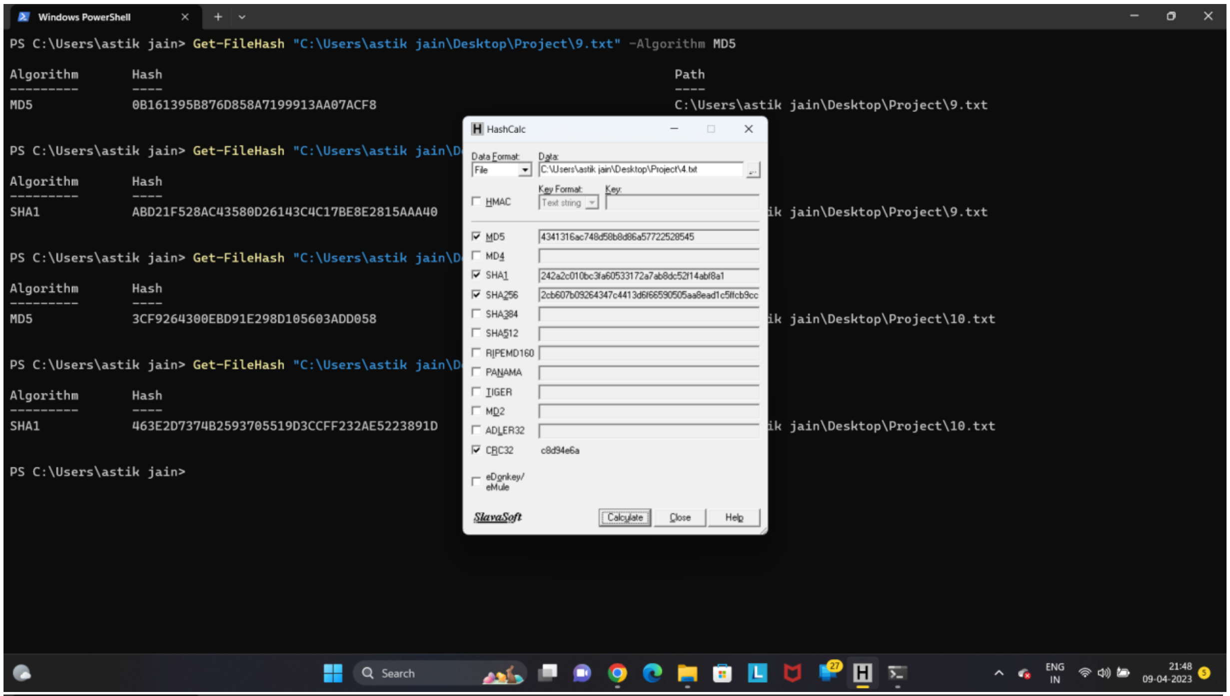

Windows PowerShell
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\Desktop\Project\9.txt" -Algorithm MD5
Algorithm Hash Path
MD5 0B161395B876D858A7199913AA07ACF8 C:\Users\astik jain\Desktop\Project\9.txt
SHA1 ABD21F528AC43580D26143C4C17BE8E2815AAA40
MD5 3CF9264300EBD91E298D105603ADD058
SHA1 463E2D7374B2593705519D3CCFF232AE5223891D
PS C:\Users\astik jain>
HashCalc
Data Format: File
Data: C:\Users\astik jain\Desktop\Project\2.txt
HMAC
Key Format: Text string
Key:
MD5 8721c58772559c4914c6d55665ccf22e
MD4
SHA1 b33e839eeead75a3f6fb12a31b4018920f54d9fc
SHA256 03db53b3b6c07bdb53ae0da0e14d1fd963db3638d7f4c6
SHA384
SHA512
RIPEMD160
PANAMA
TIGER
MD2
ADLER32
CRC32 1e80ad77
eDonkey/ eMule
SlavaSoft
Calculate
Close
Help
Search
ENG IN
21:48 09-04-2023

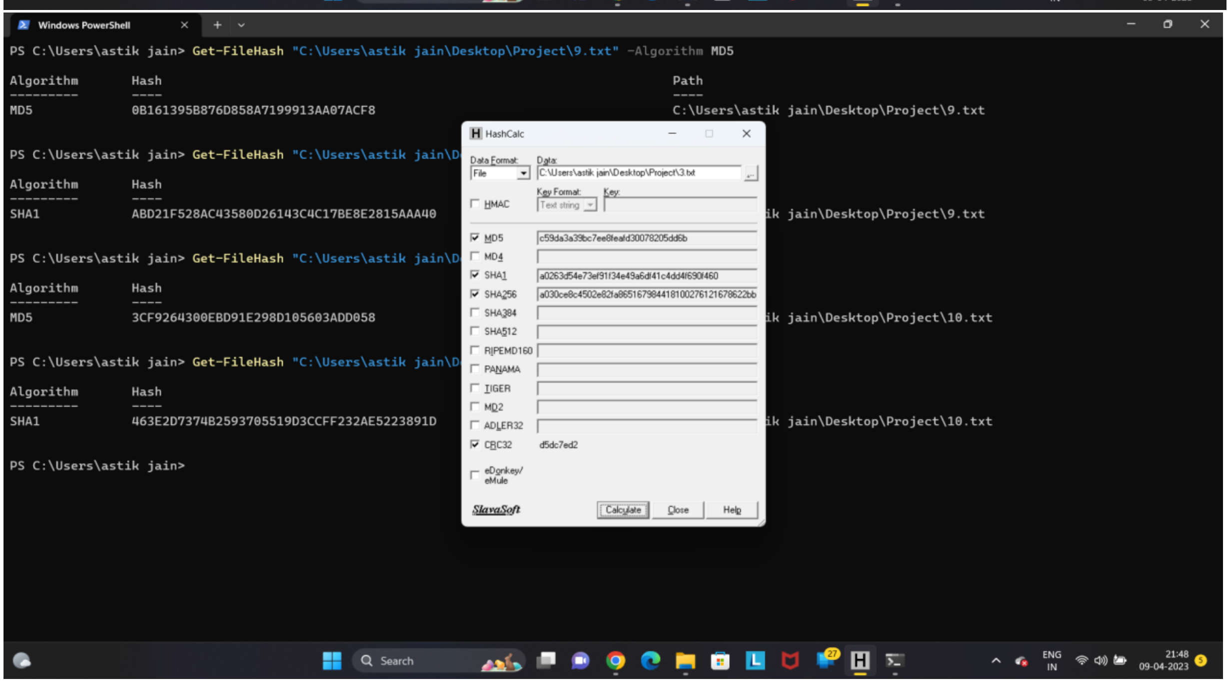







Windows PowerShell
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\Desktop\Project\9.txt" -Algorithm MD5
Algorithm Hash Path
MD5 0B161395B876D858A7199913AA07ACF8 C:\Users\astik jain\Desktop\Project\9.txt
SHA1 ABD21F528AC43580D26143C4C17BE8E2815AAA40
MD5 3CF9264300EBD91E298D105603ADD058
SHA1 463E2D7374B2593705519D3CCFF232AE5223891D
PS C:\Users\astik jain>
HashCalc
CRC32 4e4d3cc4
SlavaSoft
Calculate
Close
Help

Windows PowerShell
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\Desktop\Project\9.txt" -Algorithm MD5
Algorithm Hash Path
MD5 0B161395B876D858A7199913AA07ACF8 C:\Users\astik jain\Desktop\Project\9.txt
SHA1 ABD21F528AC43580D26143C4C17BE8E2815AAA40
MD5 3CF9264300EBD91E298D105603ADD058
SHA1 463E2D7374B2593705519D3CCFF232AE5223891D
PS C:\Users\astik jain>
HashCalc
CRC32 8511ef61
SlavaSoft
Calculate
Close
Help

Windows PowerShell
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\Desktop\Project\9.txt" -Algorithm MD5
Algorithm Hash Path
MD5 0B161395B876D858A7199913AA07ACF8 C:\Users\astik jain\Desktop\Project\9.txt
SHA1 ABD21F528AC43580D26143C4C17BE8E2815AAA40
MD5 3CF9264300EBD91E298D105603ADD058
SHA1 463E2D7374B2593705519D3CCFF232AE5223891D
PS C:\Users\astik jain>
HashCalc
CRC32 03859dcf
SlavaSoft
Calculate
Close
Help


Windows PowerShell
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\Desktop\Project\9.txt" -Algorithm MD5
Algorithm Hash Path
MD5 0B161395B876D858A7199913AA07ACF8 C:\Users\astik jain\Desktop\Project\9.txt
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\D
SHA1 ABD21F528AC43580D26143C4C17BE8E2815AAA40
ik jain\Desktop\Project\9.txt
MD5 3CF9264300EBD91E298D105603ADD058
ik jain\Desktop\Project\10.txt
SHA1 463E2D7374B2593705519D3CCFF232AE5223891D
ik jain\Desktop\Project\10.txt
PS C:\Users\astik jain>
HashCalc
Data Format: File
Data: C:\Users\astik jain\Desktop\Project\4.txt
HMAC
Key Format: Text string
Key:
MD5 4341316ac748d58b8d86a57722528545
MD4
SHA1 242a2c010bc3fa60533172a7ab8dc52f14abf8a1
SHA256 2cb607b09264347c4413d6f66590505aa8ead1c5ffcb9cc
SHA384
SHA512
RIPEMD160
PANAMA
TIGER
MD2
ADLER32
CRC32 c8d94e6a
eDonkey/ eMule
SlavaSoft
Calculate
Close
Help
Search
ENG IN
21:48
09-04-2023


Windows PowerShell
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\Desktop\Project\9.txt" -Algorithm MD5
Algorithm Hash Path
MD5 0B161395B876D858A7199913AA07ACF8 C:\Users\astik jain\Desktop\Project\9.txt
PS C:\Users\astik jain> Get-FileHash "C:\Users\astik jain\D
SHA1 ABD21F528AC43580D26143C4C17BE8E2815AAA40
ik jain\Desktop\Project\9.txt
MD5 3CF9264300EBD91E298D105603ADD058
ik jain\Desktop\Project\10.txt
SHA1 463E2D7374B2593705519D3CCFF232AE5223891D
ik jain\Desktop\Project\10.txt
PS C:\Users\astik jain>
HashCalc
Data Format: File
Data: C:\Users\astik jain\Desktop\Project\3.txt
HMAC
Key Format: Text string
Key:
MD5 c59da3a39bc7ee8feafd30078205dd6b
MD4
SHA1 a0263d54e73ef91f34e49a6df41c4dd4f690f460
SHA256 a030ce8c4502e82fa865167984418100276121678622bb
SHA384
SHA512
RIPEMD160
PANAMA
TIGER
MD2
ADLER32
CRC32 d5dc7ed2
eDonkey/ eMule
SlavaSoft
Calculate
Close
Help
Search
ENG IN
21:48
09-04-2023

Reference

- https://www.howtogeek.com/67241/htg-explains-what-are-md5-sha-1-hashes-and-how-do-i-check-them/
- https://www.slavasoft.com/zip/hashcalc.zip
- https://www.texttool.com/md5-online
- www.wikipidea.org
- www.google.com
- Window powershell