Journal scientific and applied research, vol. 27, 2024
International Journal

ISSN 1314-6289 (Print)
ISSN 2815-4622 (Online)

Original Contribution

PRACTICAL APPLICATIONS OF HASH FUNCTIONS MD5, SHA-1, AND SHA-256 USING VARIOUS SOFTWARE TOOLS TO VERIFY THE INTEGRITY OF FILES

Petar Kr. Boyanov

COMMUNICATION AND COMPUTER TECHNOLOGIES, FACULTY OF TECHNICAL SCIENCES, KONSTANTIN PRESLAVSKY UNIVERSITY OF SHUMEN, SHUMEN 9712, 115, UNIVERSITETSKA STR., E-MAIL: petar.boyanov@shu.bg

ABSTRACT: *In this scientific paper hash functions such as MD5, SHA-1, and SHA-256 are widely utilized in the field of cryptography to verify file integrity, confirming that files remain unaltered and free from corruption. The practical application of the hash functions MD5, SHA-1, AND SHA-256 through five widely used software tools: 7z, PowerShell, BullZip MD5 Calculator, HashCalc and CrypTool 1.4.40 is made. The analysis focuses on each tool's effectiveness in calculating and comparing hash values, emphasizing their strengths and weaknesses in ensuring data integrity. By investigating the functionality and applications of each tool, this scientific study provides important perspectives on the practical significance of MD5, SHA-1, and SHA-256 for maintaining the authenticity and reliability of digital files.*

KEY WORDS: *Algorithms, Calculation, CrypTool, Hash, HashCalc, Integrity, MD5, MD5 Calculator, Privacy, Security, SHA-1, SHA-256, 7z.*

1. Introduction

Hash functions such as MD5, SHA-1, and SHA-256 [31,35] are widely utilized in the field of cryptography to verify files, text and hex strings integrity [1,4,5,6,7,15,16,21,22,33,36]. This scientific paper investigates the practical application of these hash functions basically across four popular software tools: PowerShell, BullZip MD5 Calculator, HashCalc, and CrypTool 1.4.40 [2,8,9,11,19,20,24,28,29,32]. Each software tool's effectiveness in calculating and comparing hash values is analyzed, highlighting its strengths and limitations for verifying the important information integrity [1,4,5,6,7,15,16,21,22,33,36]. By examining each tool's functionality and use cases, this scientific study provides insights into the applicability of MD5, SHA-1, and SHA-256 in real-world scenarios for ensuring the authenticity and reliability of digital files. Another important task of hash functions is checking that the important and confidential information [1,4,5,6,7,15,16,21,22,33,36] has not been compromised and damaged.

In the digital era, the need for secure data verification has become critical, especially when receiving and transmitting data over unsecured computer networks. Files, text and hex strings integrity verification consists of confirming that a file has not been modified, either maliciously or due to transfer anomaly errors [1,4,5,6,7,15,16,21,22,33,36]. The most used Cryptographic hash functions, such as MD5, SHA-1, and SHA-256, produce unique "fingerprints" for files, text and hex strings, which are widely used to check whether whole data remains unmodified and intact from their original form. The MD5 (Message Digest Algorithm 5) [1,4,5,6,7,15,16,21,22,33,36] function generates a 128-bit hash value and it is mainly used for files, text and hex strings integrity verification and password storage. It is rapid and lightweight but also is vulnerable to collision attacks, meaning two different inputs can generate the same hash. The hash algorithm SHA-1 produce a 160-bit hash value and it was developed as a more secure alternative to MD5 [1,4,5,6,7,15,16,21,22,33,36]. Mainly, it finds application in Digital authentication marks and secure file transfer protocols (SFTP). As with the MD5 algorithm, this one is also vulnerable to collision attacks, which limit its usage in high-security software environments. The hash algorithm SHA-256 [1,31,35] forms a 256-bit hash value and it is regarded as very secure, making it ideal for applications like encrypted verification tokens, digital trust certificates and cryptographic ledger system. SHA-256 imposes a higher processing demand than the other two hash algorithms.

The conducted experiments in this scientific paper that aim to verify the integrity of important and confidential information without the host's permission is considered as a crime and, if proven, is punishable to the full extent of the law of the respective country [4,5]. Everything illustrated and explained in this scientific paper is for research work and educational purposes and the author is not responsible in cases of abuse.

2. Related work

These scientific works [1,4,5,6,7,15,16,21,22,33,36] collectively explore various aspects of implementation of the hash functions MD5 [2,8,9,11,19,20,24,28,29,32], SHA-1 [31,35], and SHA-256 using various software tools to verify the integrity of files [19,23,26,27,34].

The handling with of hash functions is also used in application of electronic platforms [25], various types of instrumental equipment for cyberattack prevention [18], specific models for accessing information resources in a secure environment and other technologies [17], net model of command and control system [13], building data center system for defense and security [12], designing and implementation of software-defined systems [14], information exchange management in multimodule multi-position security systems [10], applications of Artificial Intelligence in e-Learning [26], information systems for crisis prevention [30], performance analysis of a mobile computer equipped with solid state disk [38], modeling and calculation of passive audio crossovers [37] and designing of stream ciphers based on random feedback shift registers [3].

3. Experiment

The scientific experiments in this paper in a specialized computer network laboratory in the Faculty of Technical Sciences of the Konstantin Preslavsky University of Shumen are made. The used operating system is Windows 10 Pro x64 version 22H2, OS build: 19045.4355 [4,5].

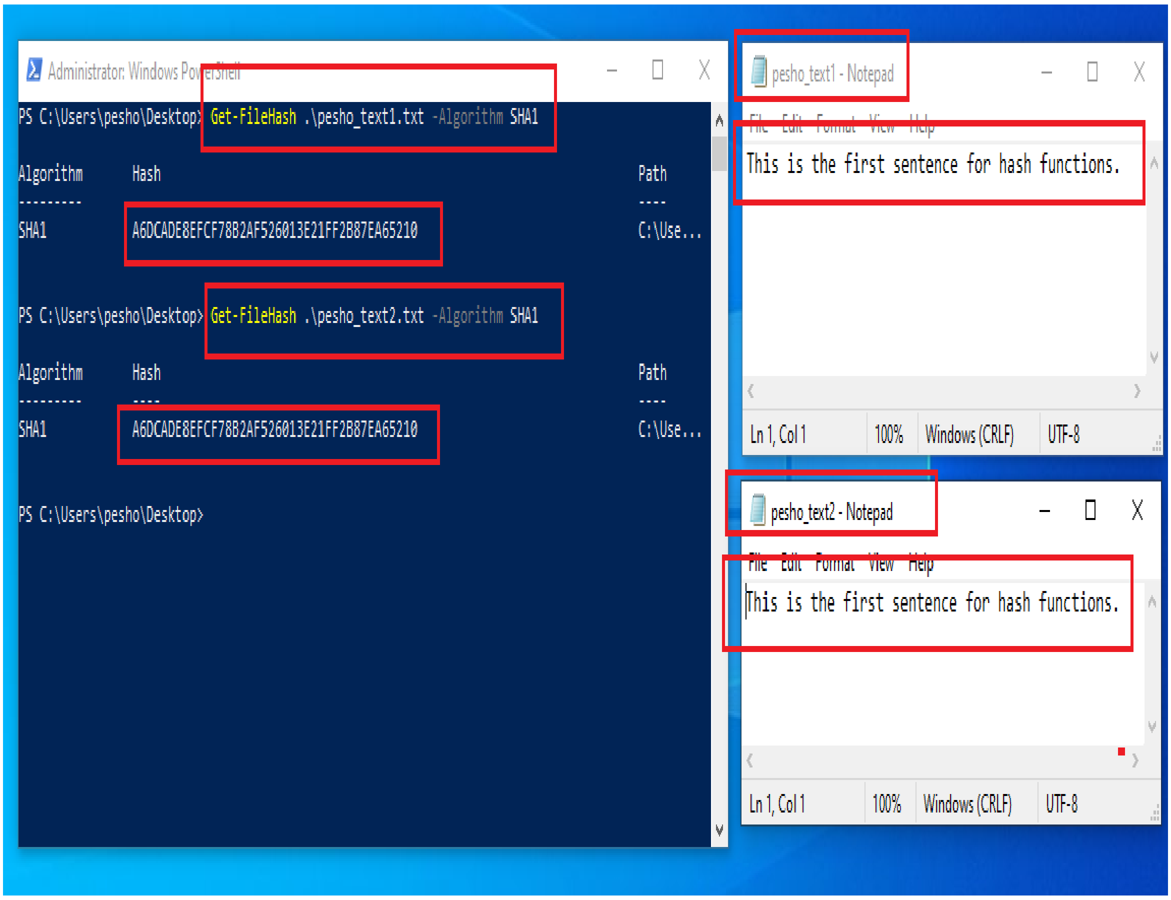
PowerShell’s cryptographic library (System.Security.Cryptography) enables hash calculation directly through command-line scripting. By utilizing the command “*Get-FileHash*”, users can generate hashes for RIPEMD160, MACTripleDES, MD5, SHA1, SHA256, SHA384, SHA512, [1,4,5,6,7,15,16,21,22,33,36] making it an excellent choice for automating file integrity verification and scripting tasks within IT workflows. In this regards, the following two text files have been created: “pesho_text1.txt” and “pesho_text2.txt”.

The both files contain the same text: “This is the first sentence for hash functions.” The generated SHA-1 [1,31,35] hash value “A6DCADE8EFC F78B2AF526013E21FF2B87EA65210” for both files is the same because they have the same text content. This is shown in fig. 1. If at the end of the sentence of the first text file a hyphen is placed instead of a period, then already the two generated hash values will be different. This is presented in fig. 2. The SHA-1 hash value for first file is “50EFA52C1D694192DFB43630251076 FA1C7B128E” and the second file remains with old SHA-1 hash value “A6DCADE8EFCF78B2AF526013E21FF2B87EA65210”.

The free of charge software tool HashCalc is a versatile tool that supports multiple hash algorithms, such as [1,4,5,6,7,15,16,21,22,33,36]:

- MD2, MD4 and MD5;
- SHA-1, SHA-256, SHA-384 and SHA-512
- RIPEMD160;
- PANAMA;
- TIGER;
- ADLER32;
- CRC32;
- eDonkey/eMule.

It offers users an extensive interface for calculating hashes and HMAC for both files and text (text and hex strings), enhancing flexibility in verifying data integrity and providing educational demonstrations [19,23,26,27,34].

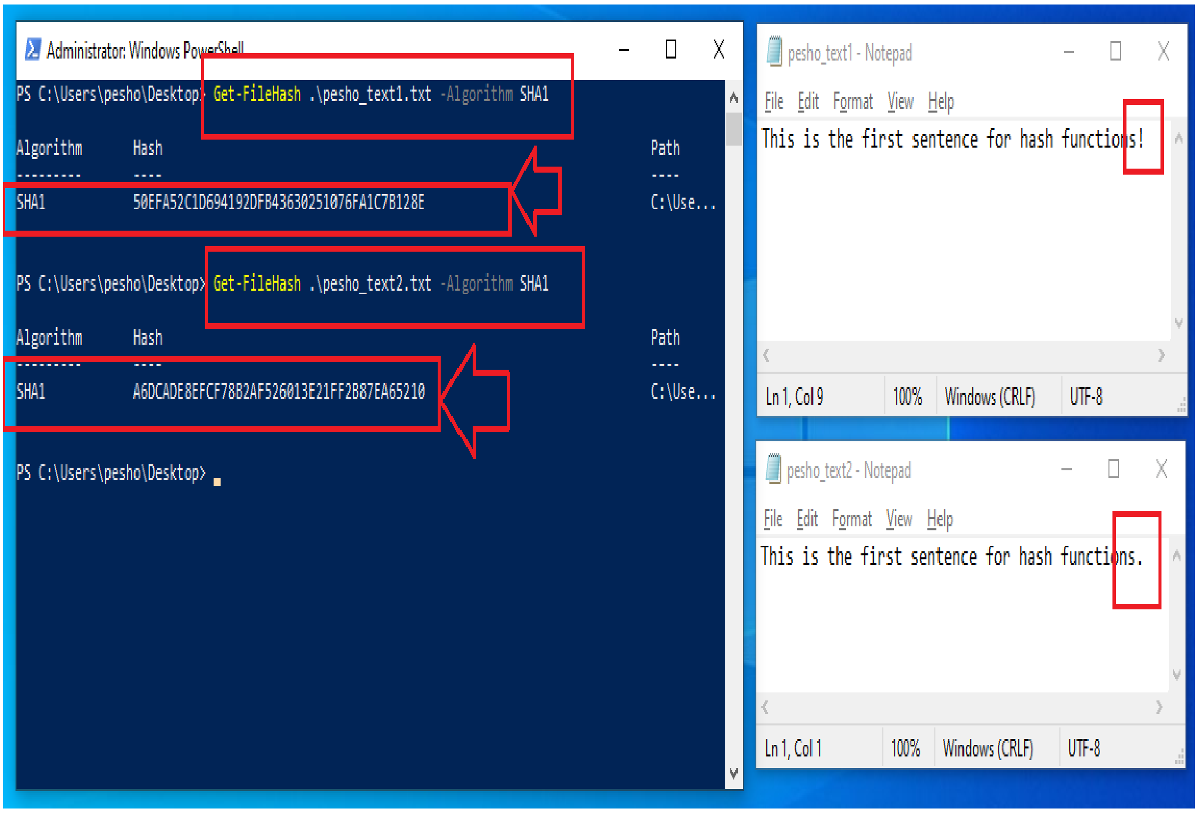


```
PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm SHA1

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA1            A6DCADE8EFCF78B2AF526013E21FF2B87EA65210                               C:\Use...

PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm SHA1

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA1            A6DCADE8EFCF78B2AF526013E21FF2B87EA65210                               C:\Use...

PS C:\Users\pesho\Desktop>
```

```
This is the first sentence for hash functions.
```

```
This is the first sentence for hash functions.
```

Fig. 1. The generated SHA-1 hash values for the same text content

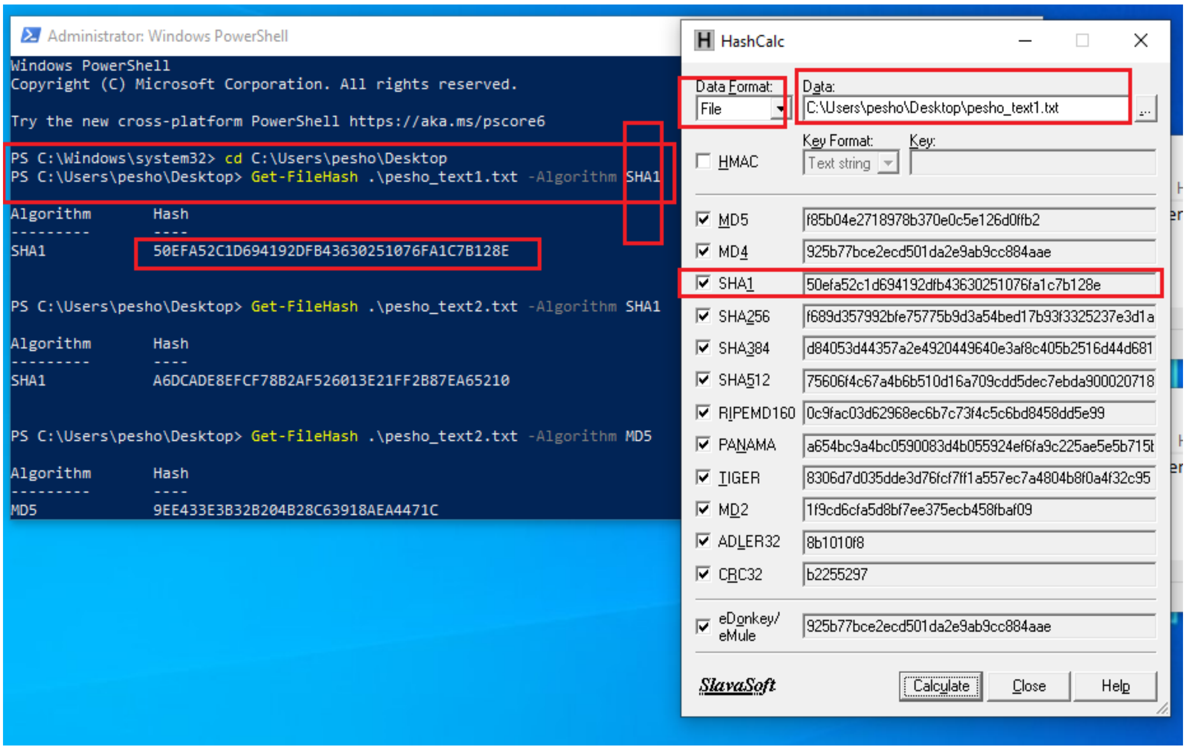


```
PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm SHA1

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA1            50EFA52C1D694192DFB43630251076FA1C7B128E                               C:\Use...

PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm SHA1

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA1            A6DCADE8EFCF78B2AF526013E21FF2B87EA65210                               C:\Use...

PS C:\Users\pesho\Desktop>
```

```
This is the first sentence for hash functions!
```

```
This is the first sentence for hash functions.
```

Fig. 2. The two different SHA-1 hash values obtained



Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> cd C:\Users\pesho\Desktop
PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm SHA1

Algorithm       Hash
---------       ----
SHA1            50EFA52C1D694192DFB43630251076FA1C7B128E


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm SHA1

Algorithm       Hash
---------       ----
SHA1            A6DCADE8EFCF78B2AF526013E21FF2B87EA65210


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm MD5

Algorithm       Hash
---------       ----
MD5             9EE433E3B32B204B28C63918AEA4471C
```

HashCalc

Data Format: File

Data: C:\Users\pesho\Desktop\pesho_text1.txt

□ HMAC Key Format: Text string Key:

☑ MD5 f85b04e2718978b370e0c5e126d0ffb2

☑ MD4 925b77bce2ecd501da2e9ab9cc884aae

☑ SHA1 50efa52c1d694192dfb43630251076fa1c7b128e

☑ SHA256 f689d357992bfe75775b9d3a54bed17b93f3325237e3d1a

☑ SHA384 d84053d44357a2e4920449640e3af8c405b2516d44d681

☑ SHA512 75606f4c67a4b6b510d16a709cdd5dec7ebda900020718

☑ RIPEMD160 0c9fac03d62968ec6b7c73f4c5c6bd8458dd5e99

☑ PANAMA a654bc9a4bc0590083d4b055924ef6fa9c225ae5e5b715

☑ TIGER 8306d7d035dde3d76fcf7ff1a557ec7a4804b8f0a4f32c95

☑ MD2 1f9cd6cfa5d8bf7ee375ecb458fbaf09

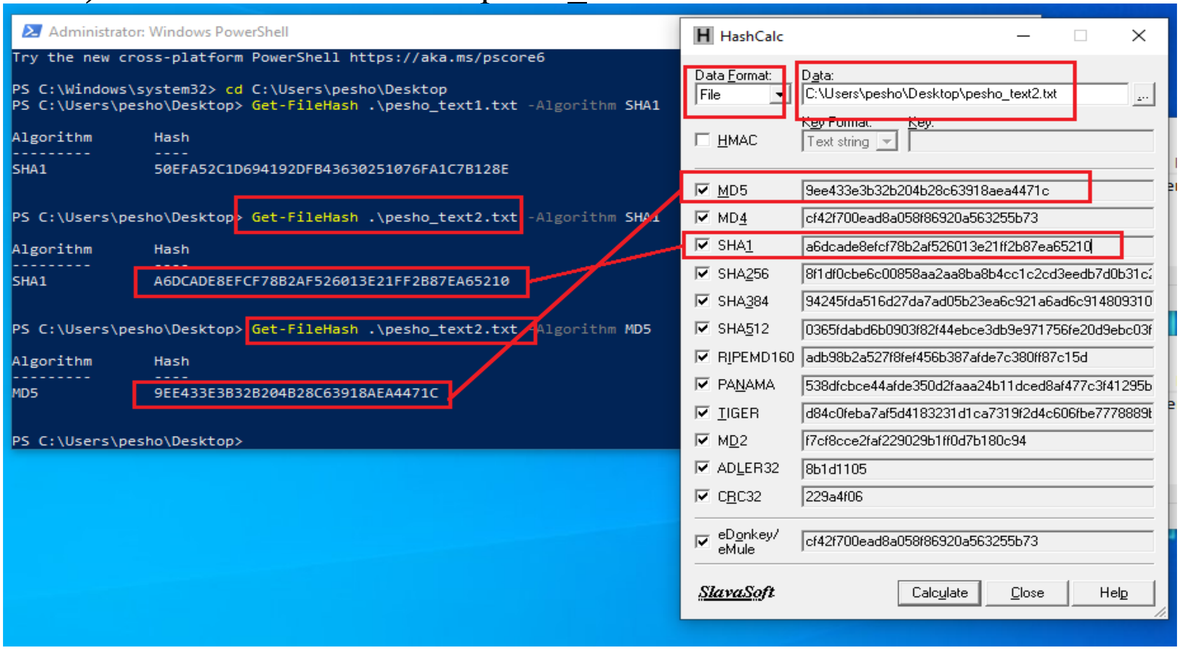
☑ ADLER32 8b1010f8

☑ CRC32 b2255297

☑ eDonkey/ eMule 925b77bce2ecd501da2e9ab9cc884aae

Fig. 3. The GUI of the program HashCalc

Fig. 3 shows that the software program HashCalc generates the same hash SHA-1 (50EFA52C1D694192DFB43630251076FA1C7B128E) values for the first file "pesho_text1.txt". Fig. 4 illustrates that the both programs PowerShell and HashCalc generate the same SHA-1 (A6DCADE8EFCF78B2AF52 6013E21FF2B87EA65210) and MD5 (9EE433E3B32B204B28C63918AEA 4471C) Hash values for the file "pesho_text2.txt".



Administrator: Windows PowerShell

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> cd C:\Users\pesho\Desktop
PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm SHA1

Algorithm       Hash
---------       ----
SHA1            50EFA52C1D694192DFB43630251076FA1C7B128E


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm SHA1

Algorithm       Hash
---------       ----
SHA1            A6DCADE8EFCF78B2AF526013E21FF2B87EA65210


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm MD5

Algorithm       Hash
---------       ----
MD5             9EE433E3B32B204B28C63918AEA4471C


PS C:\Users\pesho\Desktop>
```

HashCalc

Data Format: File

Data: C:\Users\pesho\Desktop\pesho_text2.txt

□ HMAC Key Format: Text string Key:

☑ MD5 9ee433e3b32b204b28c63918aea4471c

☑ MD4 cf42f700ead8a058f86920a563255b73

☑ SHA1 a6dcade8efcf78b2af526013e21ff2b87ea65210

☑ SHA256 8f1df0cbe6c00858aa2aa8ba8b4cc1c2cd3eedb7d0b31c

☑ SHA384 94245fda516d27da7ad05b23ea6c921a6ad6c914809310

☑ SHA512 0365fdabd6b0903f82f44ebce3db9e971756fe20d9ebc03f

☑ RIPEMD160 adb98b2a527f8fef456b387afde7c380ff87c15d

☑ PANAMA 538dfcbce44afde350d2faaa24b11dced8af477c3f41295b

☑ TIGER d84c0feba7af5d4183231d1ca7319f2d4c606fbe7778889

☑ MD2 f7cf8cce2faf229029b1ff0d7b180c94

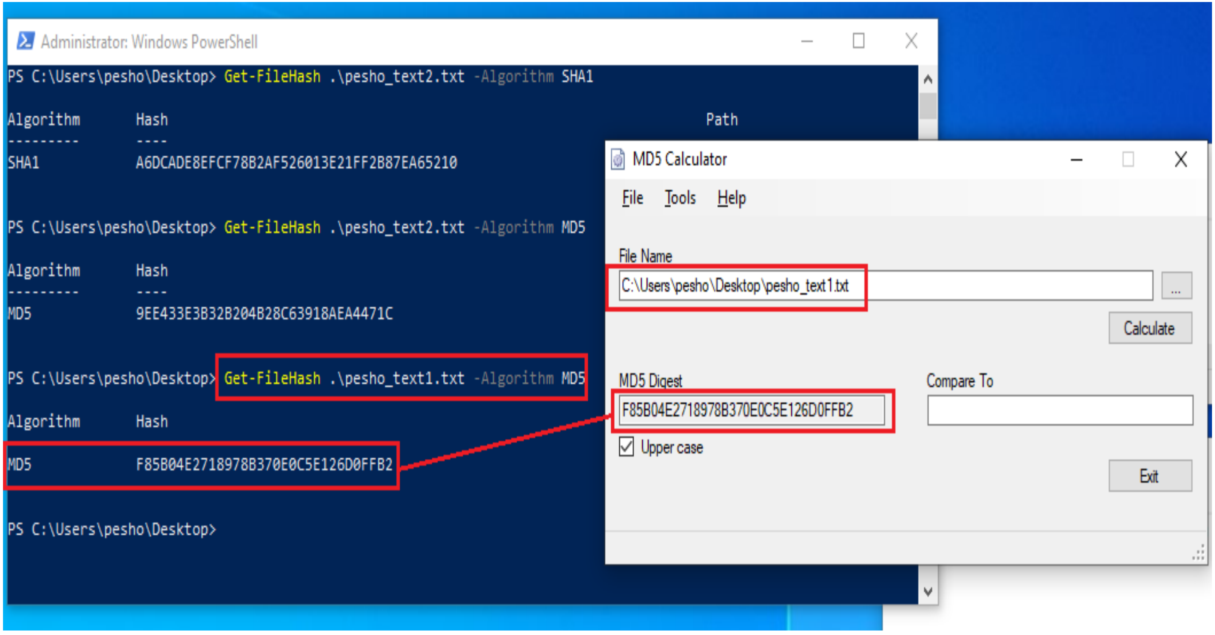
☑ ADLER32 8b1d1105

☑ CRC32 229a4f06

☑ eDonkey/ eMule cf42f700ead8a058f86920a563255b73

Fig. 4. The generated SHA-1 and MD-5 hash values for the file "pesho_text2.txt"

BullZip MD5 Calculator is a simple, GUI-based tool created specifically for generating MD5 hash values. It is primarily used for basic file integrity checks, enabling users to confirm if files have been modified by comparing MD5 hashes [1,4,5,6,7,15,16,21,22,33,36].



```
PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm SHA1

Algorithm       Hash                                                                   Path
---------       ----
SHA1            A6DCADE8EFCF78B2AF526013E21FF2B87EA65210


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm MD5

Algorithm       Hash
---------       ----
MD5             9EE433E3B32B204B28C63918AEA4471C


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm MD5

Algorithm       Hash

MD5             F85B04E2718978B370E0C5E126D0FFB2


PS C:\Users\pesho\Desktop>
```

MD5 Calculator

File Name: C:\Users\pesho\Desktop\pesho_text1.txt

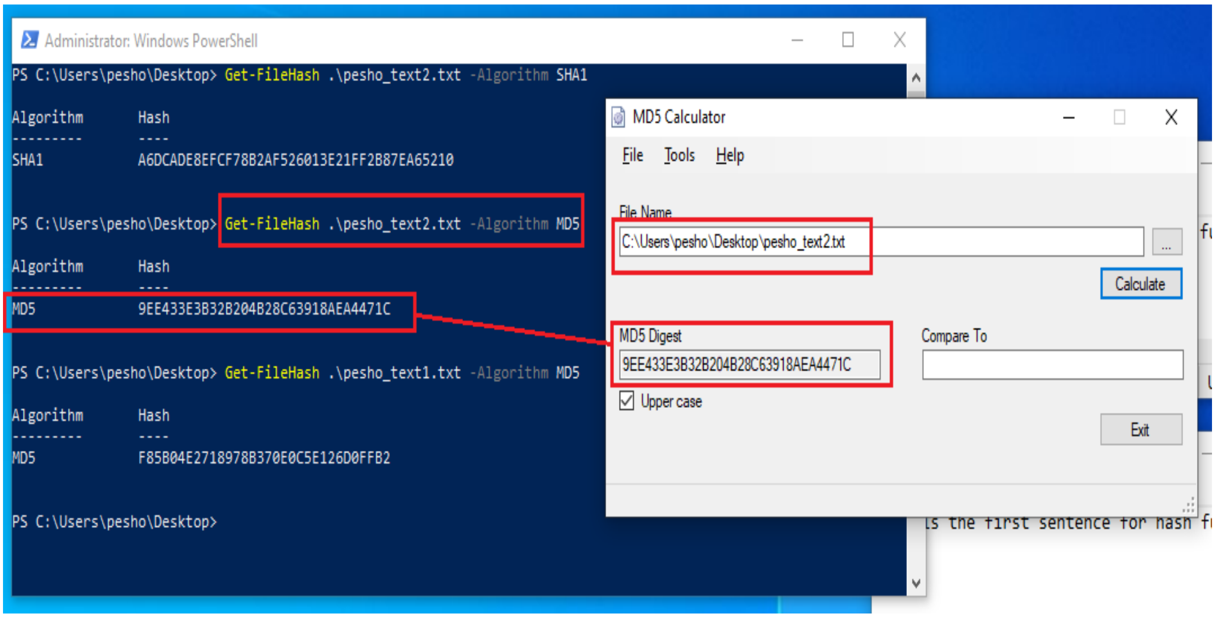
MD5 Digest: F85B04E2718978B370E0C5E126D0FFB2

Compare To:

☑ Upper case

Fig. 5. The calculated MD5 hash value for the file “pesho_text1.txt”

The calculated MD5 hash value (F85B04E2718978B370E0C5E126D0 FFB2) of the file “pesho_text1.txt” in fig. 5 via the program MD5 Calculator is presented. Fig. 6 shows the generated MD5 hash value (9EE433E3B32B2 04B28C63918AEA4471C) of file “pesho_text2.txt”. It is found that the program MD5 calculator calculates only MD5 hashes.



```
PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm SHA1

Algorithm       Hash
---------       ----
SHA1            A6DCADE8EFCF78B2AF526013E21FF2B87EA65210


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm MD5

Algorithm       Hash
---------       ----
MD5             9EE433E3B32B204B28C63918AEA4471C


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm MD5

Algorithm       Hash
---------       ----
MD5             F85B04E2718978B370E0C5E126D0FFB2


PS C:\Users\pesho\Desktop>
```

MD5 Calculator

File Name: C:\Users\pesho\Desktop\pesho_text2.txt

MD5 Digest: 9EE433E3B32B204B28C63918AEA4471C

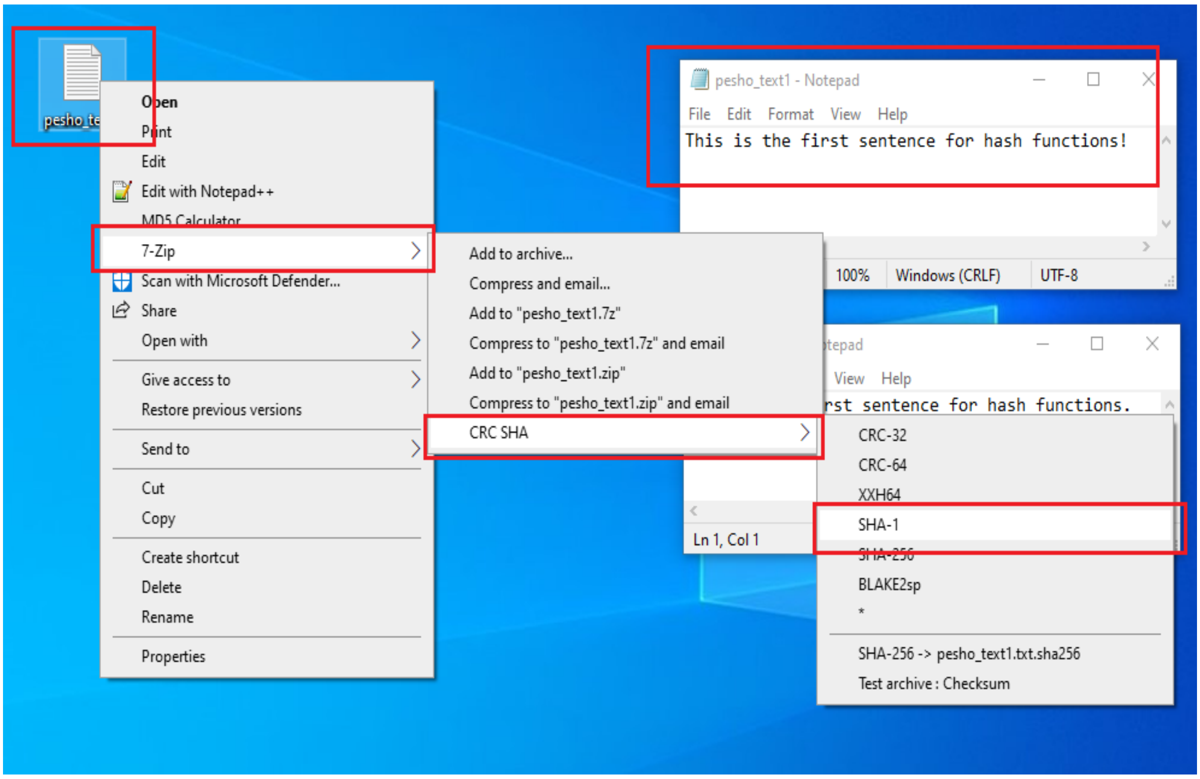
Compare To:

☑ Upper case

Administrator: Windows PowerShell
File
Tools
Help
...
Calculate
Exit
is the first sentence for hash fi

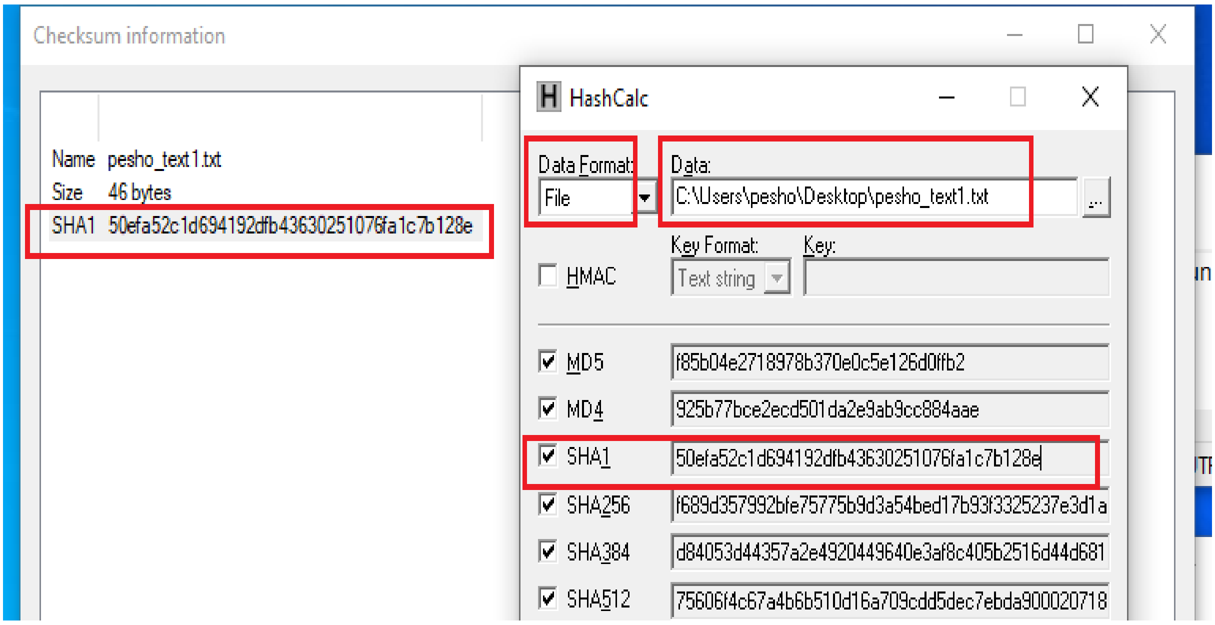
Fig. 6. The calculated MD5 hash value for the file “pesho_text2.txt”

7-Zip is a file archiving tool known for its high compression efficiency [19,23,26,27,34]. Through this program the hash value can also be calculated. It calculates the following algorithms: CRC-32, CRC-64, XXH64, SHA-1, SHA-256, and BLAKE2sp [1,4,5,6,7,15,16,21,22,33,36]. In fig. 7 the steps for generating the SHA-1 hash value of the file “pesho_text1.txt” are shown.



```
This is the first sentence for hash functions!
```

Fig. 7. The steps for generating the SHA-1 hash value of the file “pesho_text1.txt”

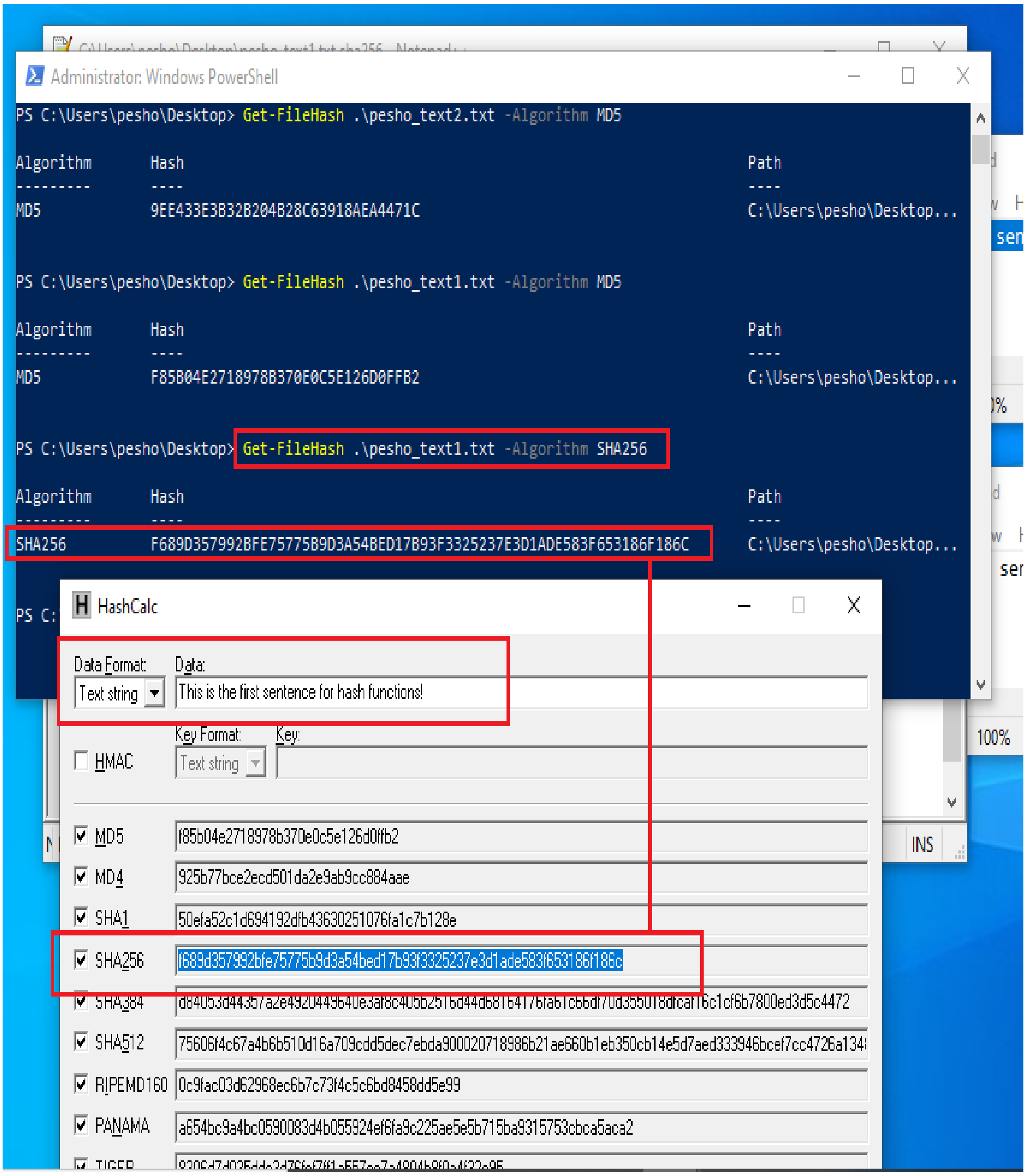


Name pesho_text1.txt
Size 46 bytes
SHA1 50efa52c1d694192dfb43630251076fa1c7b128e

Data Format: File
Data: C:\Users\pesho\Desktop\pesho_text1.txt
Key Format: Text string
Key:
□ HMAC
☑ MD5 f85b04e2718978b370e0c5e126d0ffb2
☑ MD4 925b77bce2ecd501da2e9ab9cc884aae
☑ SHA1 50efa52c1d694192dfb43630251076fa1c7b128e
☑ SHA256 f689d357992bfe75775b9d3a54bed17b93f3325237e3d1a
☑ SHA384 d84053d44357a2e4920449640e3af8c405b2516d44d681
☑ SHA512 75606f4c67a4b6b510d16a709cdd5dec7ebda900020718

Fig. 8. The generated SHA-1 hash value with the program 7z

The generated SHA-1 (50EFA52C1D694192DFB43630251076FA1C7B128E) hash value with the program 7z completely matches the value obtained with the other program HashCalc. This is shown in fig. 8.



```
PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm MD5

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
MD5             9EE433E3B32B204B28C63918AEA4471C                                       C:\Users\pesho\Desktop...


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm MD5

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
MD5             F85B04E2718978B370E0C5E126D0FFB2                                       C:\Users\pesho\Desktop...


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm SHA256

Algorithm       Hash                                                                   Path
---------       ----                                                                   ----
SHA256          F689D357992BFE75775B9D3A54BED17B93F3325237E3D1ADE583F653186F186C       C:\Users\pesho\Desktop...


PS C:
```

Data Format: Text string
Data: This is the first sentence for hash functions!

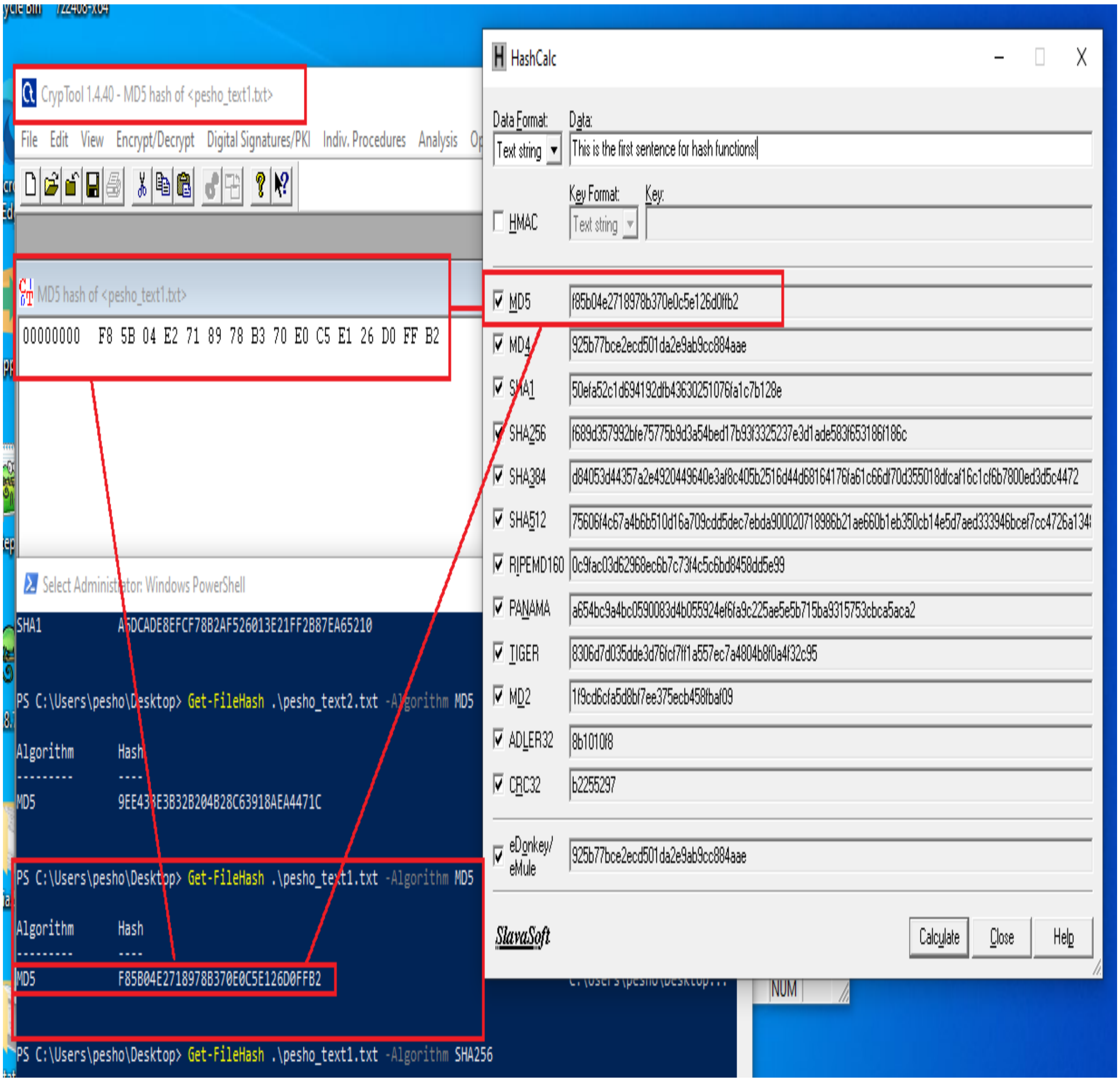
☐ HMAC Key Format: Text string Key:

☑ MD5 f85b04e2718978b370e0c5e126d0ffb2
☑ MD4 925b77bce2ecd501da2e9ab9cc884aae
☑ SHA1 50efa52c1d694192dfb43630251076fa1c7b128e
☑ SHA256 f689d357992bfe75775b9d3a54bed17b93f3325237e3d1ade583f653186f186c
☑ SHA384 d84053d44357a2e49204495640e3af8c405b2516d44d6816417bfab1c6bdf70d355018dfcaf16c1cf6b7800ed3d5c4472
☑ SHA512 75606f4c67a4b6b510d16a709cdd5dec7ebda900020718986b21ae660b1eb350cb14e5d7aed333946bcef7cc4726a134
☑ RIPEMD160 0c9fac03d62968ec6b7c73f4c5c6bd8458dd5e99
☑ PANAMA a654bc9a4bc0590083d4b055924ef6fa9c225ae5e5b715ba9315753cbca5aca2
☑ TIGER [illegible]

Fig. 9. The generated SHA-256 hash values for the text string “This is the first sentence for hash functions!” and file “pesho_text1.txt”

The next task, which is shown in fig. 9, is related to the calculation of the SHA-256 [1,31,35] hash value of the text string contained in the file “pesho_text1.txt”. As a result, it is found that if a hash is calculated on a file or its entire contents, the generated hash value would be the same in both cases.

The software program Cryptool version 1.4.40 [2,8,9,11,19,20,24,28,29,32] is programmed to generate the following hashes: MD2, MD4, DM5, SHA-1, SHA-256, SHA-512 and RIPEMD-160. In fig. 10 the generated MD5 hash value (F85B04E2718978B370E0C5E126D0FFB2) of the file "pesho_text1.txt" with the software tools CrypTool and PowerShell, and the generated MD5 hash value of the text string "This is the first sentence for hash functions!" is presented.


CrypTool 1.4.40 - MD5 hash of <pesho_text1.txt>
File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Op
MD5 hash of <pesho_text1.txt>
Select Administrator: Windows PowerShell
HashCalc
SlavaSoft
Calculate
Close
Help
NUM

```
00000000  F8 5B 04 E2 71 89 78 B3 70 E0 C5 E1 26 D0 FF B2
```

```
SHA1          A5DCADE8EFCF78B2AF526013E21FF2B87EA65210


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm MD5

Algorithm     Hash
---------     ----
MD5           9EE433E3B32B204B28C63918AEA4471C


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm MD5

Algorithm     Hash
---------     ----
MD5           F85B04E2718978B370E0C5E126D0FFB2


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm SHA256
```

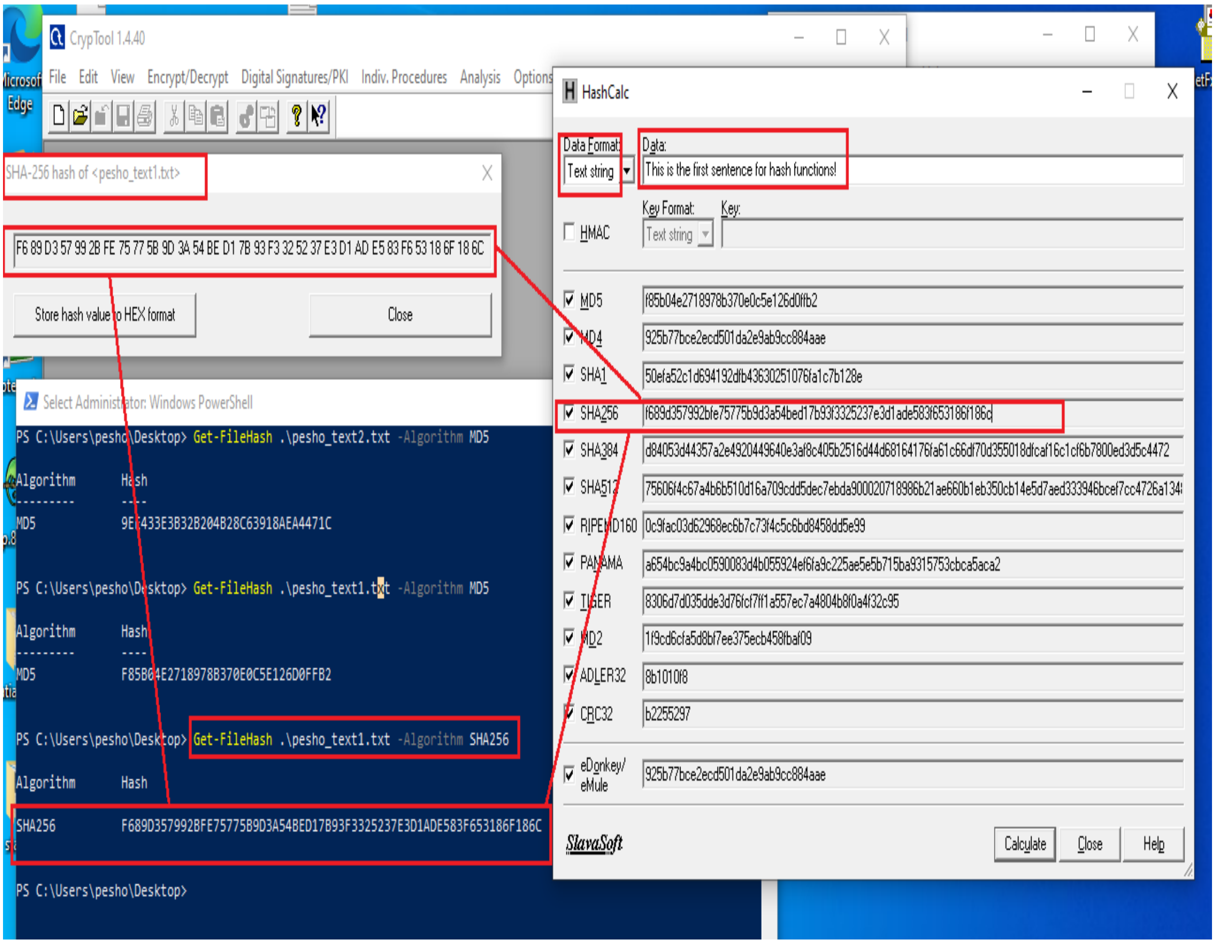
Data Format: Text string
Data: This is the first sentence for hash functions!

□ HMAC   Key Format: Text string   Key:

☑ MD5 f85b04e2718978b370e0c5e126d0ffb2
☑ MD4 925b77bce2ecd501da2e9ab9cc884aae
☑ SHA1 50efa52c1d694192dfb43630251076fa1c7b128e
☑ SHA256 f689d357992bfe75775b9d3a54bed17b93f3325237e3d1ade583f653186f186c
☑ SHA384 d84053d44357a2e4920449640e3af8c405b2516d44d68164176fa61c66df70d355018dfcaf16c1cf6b7800ed3d5c4472
☑ SHA512 75606f4c67a4b6b510d16a709cdd5dec7ebda900020718986b21ae660b1eb350cb14e5d7aed333946bcef7cc4726a134
☑ RIPEMD160 0c9fac03d62968ec6b7c73f4c5c6bd8458dd5e99
☑ PANAMA a654bc9a4bc0590083d4b055924ef6fa9c225ae5e5b715ba9315753cbca5aca2
☑ TIGER 8306d7d035dde3d76fcf7ff1a557ec7a4804b8f0a4f32c95
☑ MD2 1f9cd6cfa5d8bf7ee375ecb458fbaf09
☑ ADLER32 8b1010f8
☑ CRC32 b2255297
☑ eDonkey/eMule 925b77bce2ecd501da2e9ab9cc884aae

Fig. 10. The generated MD5 hash value of the file "pesho_text1.txt" and with the programs CrypTool, HashCalc and PowerShell

In fig. 11 the generated SHA-256 hash value (F689D357992BFE7577 5B9D3A54BED17B93F3325237E3D1ADE583F653186F186C) of the file "pesho_text1.txt" with the software tools CrypTool and PowerShell, and the generated SHA-256 hash value of the text string "This is the first sentence for hash functions!" is presented.


CrypTool 1.4.40
File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options
Store hash value to HEX format
Close
Select Administrator: Windows PowerShell
HashCalc
Calculate
Close
Help
SlavaSoft

SHA-256 hash of <pesho_text1.txt>

F6 89 D3 57 99 2B FE 75 77 5B 9D 3A 54 BE D1 7B 93 F3 32 52 37 E3 D1 AD E5 83 F6 53 18 6F 18 6C

```
PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text2.txt -Algorithm MD5

Algorithm       Hash
---------       ----
MD5             9EF433E3B32B204B28C63918AEA4471C


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm MD5

Algorithm       Hash
---------       ----
MD5             F85B04E2718978B370E0C5E126D0FFB2


PS C:\Users\pesho\Desktop> Get-FileHash .\pesho_text1.txt -Algorithm SHA256

Algorithm       Hash

SHA256          F689D357992BFE75775B9D3A54BED17B93F3325237E3D1ADE583F653186F186C


PS C:\Users\pesho\Desktop>
```

Data Format: Text string
Data: This is the first sentence for hash functions!

☐ HMAC Key Format: Text string Key:

☑ MD5 f85b04e2718978b370e0c5e126d0ffb2
☑ MD4 925b77bce2ecd501da2e9ab9cc884aae
☑ SHA1 50efa52c1d694192dfb43630251076fa1c7b128e
☑ SHA256 f689d357992bfe75775b9d3a54bed17b93f3325237e3d1ade583f653186f186c
☑ SHA384 d84053d44357a2e4920449640e3af8c405b2516d44d68164176fa61c66df70d355018dfcaf16c1cf6b7800ed3d5c4472
☑ SHA512 75606f4c67a4b6b510d16a709cdd5dec7ebda900020718986b21ae660b1eb350cb14e5d7aed333946bcef7cc4726a134
☑ RIPEMD160 0c9fac03d62968ec6b7c73f4c5c6bd8458dd5e99
☑ PANAMA a654bc9a4bc0590083d4b055924ef6fa9c225ae5e5b715ba9315753cbca5aca2
☑ TIGER 8306d7d035dde3d76fcf7ff1a557ec7a4804b8f0a4f32c95
☑ MD2 1f9cd6cfa5d8bf7ee375ecb458fbaf09
☑ ADLER32 8b1010f8
☑ CRC32 b2255297
☑ eDonkey/eMule 925b77bce2ecd501da2e9ab9cc884aae

Fig. 11. The generated SHA-256 hash value of the file “pesho_text1.txt” and with the programs CrypTool, HashCalc and PowerShell

4. Results

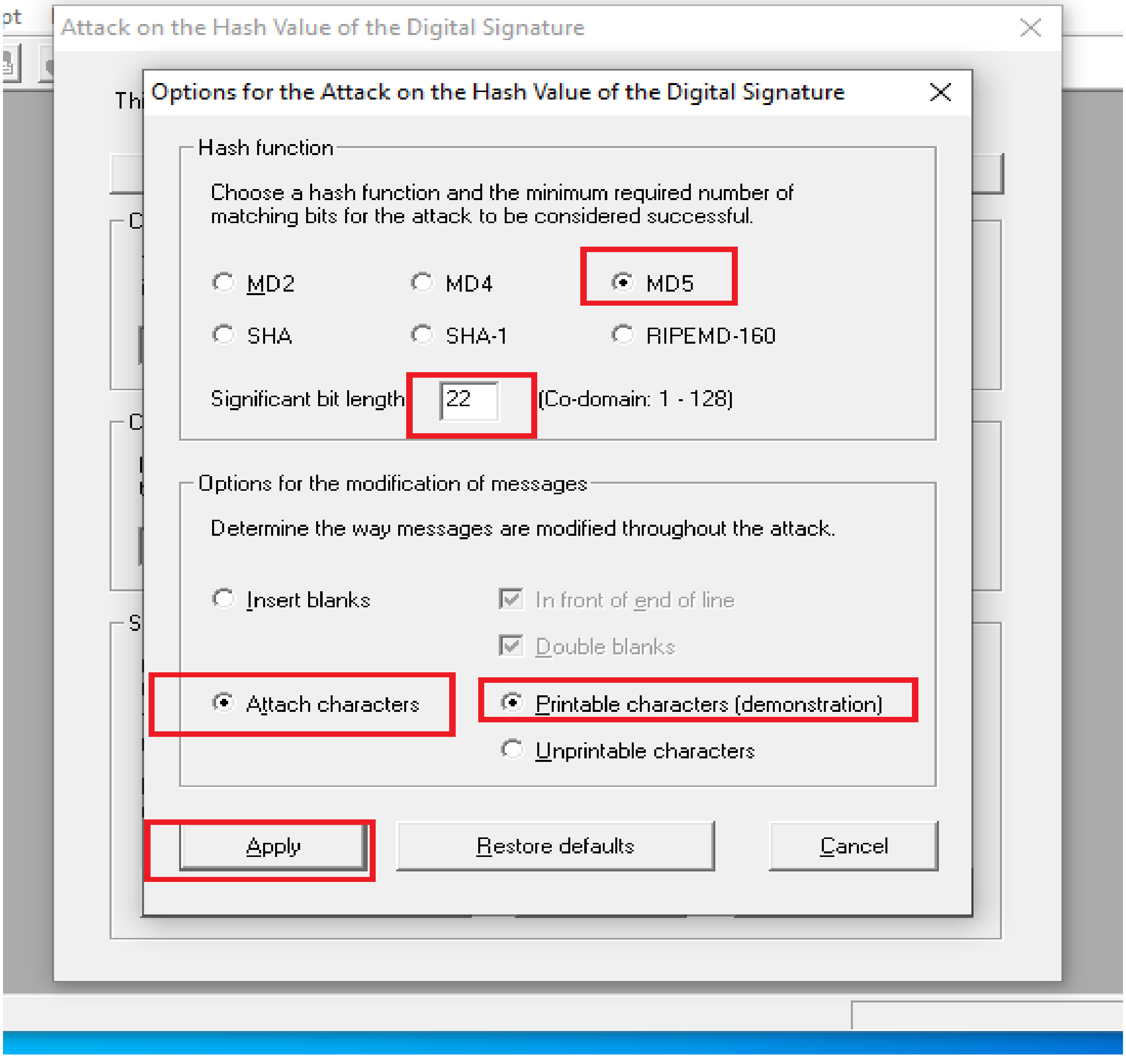
The software tool HashCalc [1,4,5,6,7,15,16,21,22,33,36] generates multiple hashes simultaneously and enables users to verify file integrity across systems that require different hash algorithms. The educators can utilize HashCalc to illustrate hash properties, such as output length and the avalanche effect, across various algorithms. HashCalc serves as a free of charge software educational tool, enabling users to see the varying hash outputs for a single input file, text and hex string, effectively showcasing the differences between hash functions and their unique output lengths.

CrypTool [2,8,9,11,19,20,24,28,29,32] is a useful resource for academic settings, providing students and researchers with a platform to explore cryptographic principles and conduct experiments with hash functions. CrypTool's capacity [2,8,9,11,19,20,24,28,29,32] to simulate hash functions makes it a valuable resource for cryptographic research and analysis, particularly in visualizing potential hash collisions with MD5 and SHA-1. CrypTool enables users to test different inputs to observe changes in hash outputs, effectively illustrating both the avalanche effect and the deterministic properties of hash functions [1,4,5,6,7,15,16,21,22,33,36].

The users can create MD5 hash for a file and check it against a known hash in order to verify the file's integrity. This method is commonly used to confirm downloaded files, by matching the computed hash with the MD5 hash value provided by the source file. Due to MD5's vulnerabilities, BullZip MD5 Calculator is not recommended for secure applications; however it is still effective for integrity checks in non-sensitive situations [19,23,26,27,34].

The Windows based command-line tool, PowerShell has the capability to automate batch hash verification for multiple files, which is advantageous for system and network administrators.

This software application CrypTool [2,8,9,11,19,20,24,28,29,32] allows the execution of a cyber-attack targeting the hash value of Electronic authentication. The chosen hash function is MD5 and the significant bit length is set to 22 bits. During the cyber-attack intercepted messages will be modified with additional characters at the end of the text message contained in the file.


Attack on the Hash Value of the Digital Signature

Options for the Attack on the Hash Value of the Digital Signature

Hash function

Choose a hash function and the minimum required number of matching bits for the attack to be considered successful.

○ MD2 ○ MD4 ● MD5

○ SHA ○ SHA-1 ○ RIPEMD-160

Significant bit length 22 (Co-domain: 1 - 128)

Options for the modification of messages

Determine the way messages are modified throughout the attack.

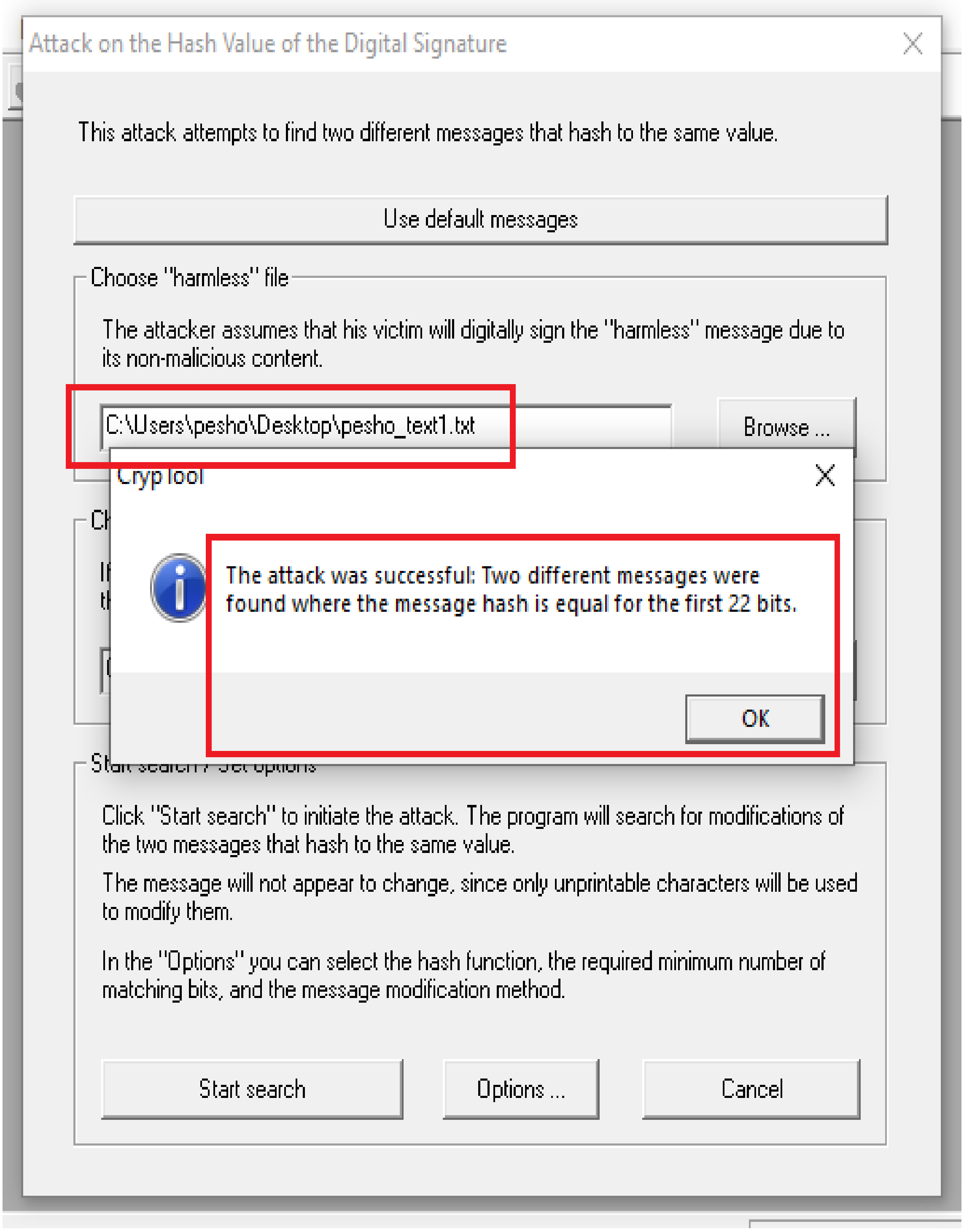
○ Insert blanks ☑ In front of end of line

☑ Double blanks

● Attach characters ● Printable characters (demonstration)

○ Unprintable characters

Fig. 12. The Cyber-attack settings



This attack attempts to find two different messages that hash to the same value.

Choose "harmless" file

The attacker assumes that his victim will digitally sign the "harmless" message due to its non-malicious content.

C:\Users\pesho\Desktop\pesho_text1.txt

The attack was successful: Two different messages were found where the message hash is equal for the first 22 bits.

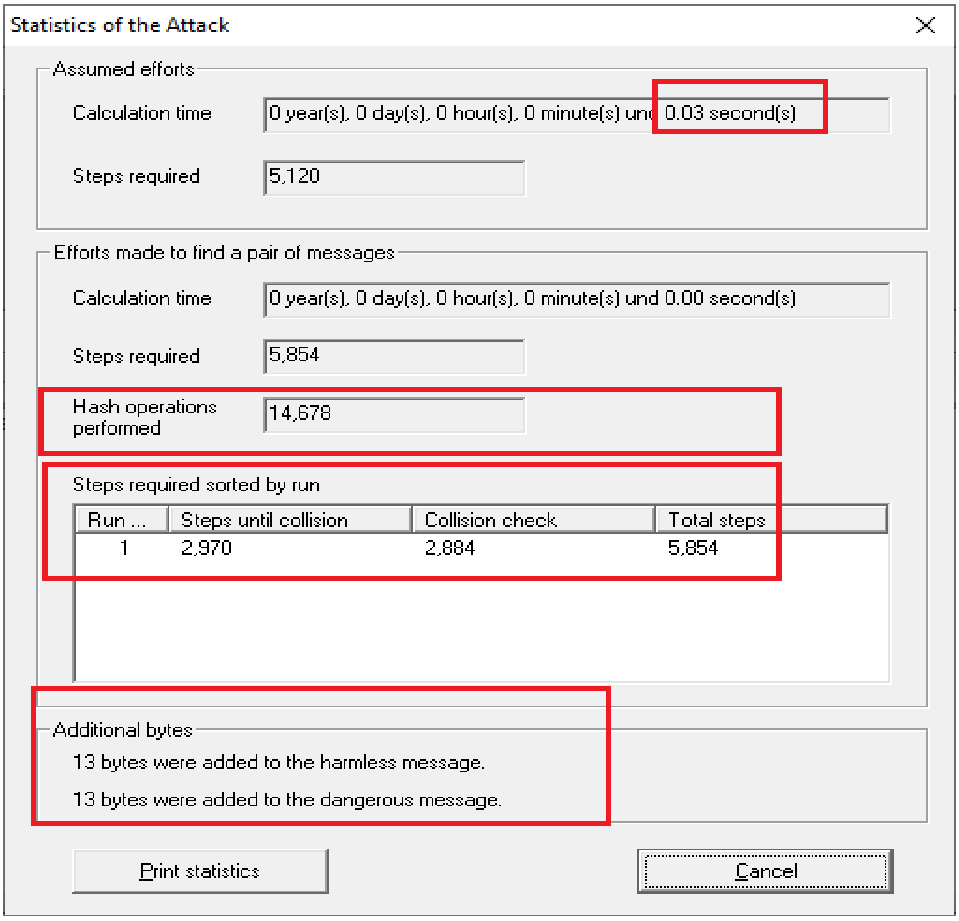
Click "Start search" to initiate the attack. The program will search for modifications of the two messages that hash to the same value.

The message will not appear to change, since only unprintable characters will be used to modify them.

In the "Options" you can select the hash function, the required minimum number of matching bits, and the message modification method.

Fig. 13. The result of the conducted experimental cyber-attack

Fig. 13 presents that the hash value is equal for the first 22 bits for the both different message files “pesho_text1.txt” and “pesho_text2.txt”. Fig. 14 and 15 show the statistics of the conducted cyber-attack. The calculation time took only 0.03 s. with 14.678 performed hash operations and 5854 total steps for the both files. Fig. 16 illustrates the adding of 13 bytes to the both text messages.



Statistics of the Attack

Assumed efforts

Calculation time: 0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.03 second(s)

Steps required: 5,120

Efforts made to find a pair of messages

Calculation time: 0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.00 second(s)

Steps required: 5,854

Hash operations performed: 14,678

Steps required sorted by run

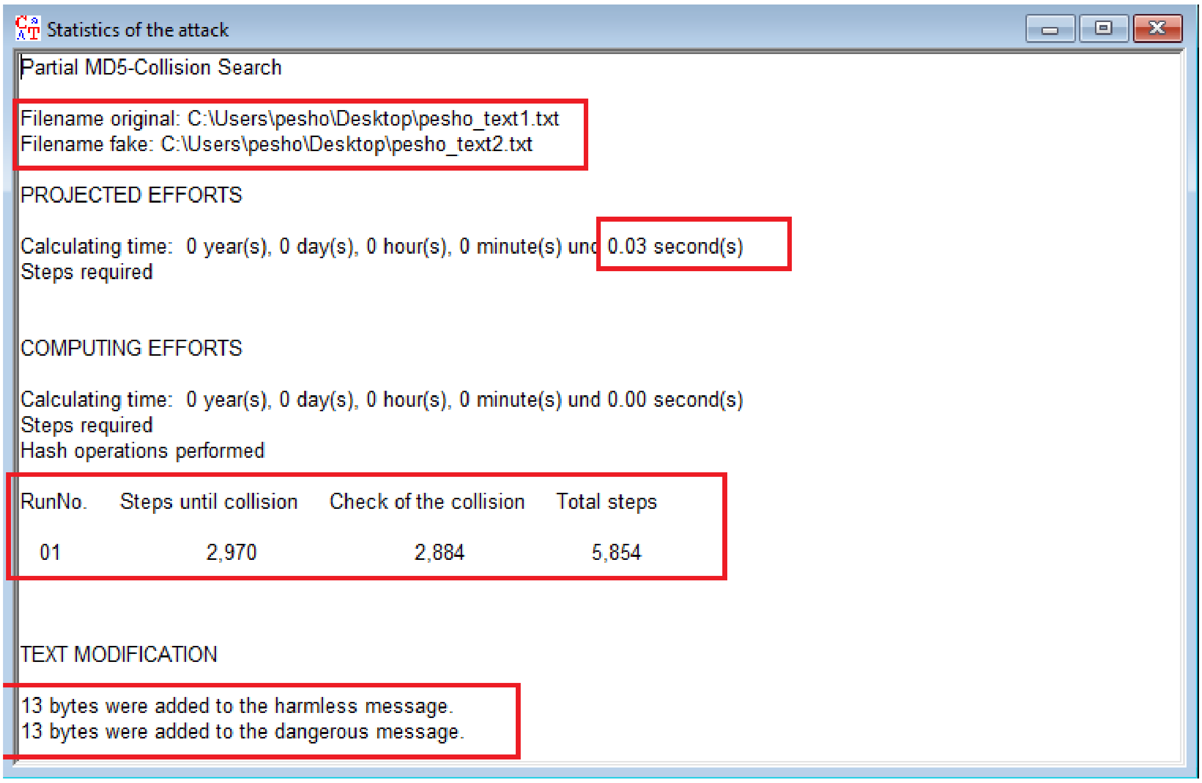
Run ...	Steps until collision	Collision check	Total steps
1	2,970	2,884	5,854

Additional bytes

13 bytes were added to the harmless message.

13 bytes were added to the dangerous message.

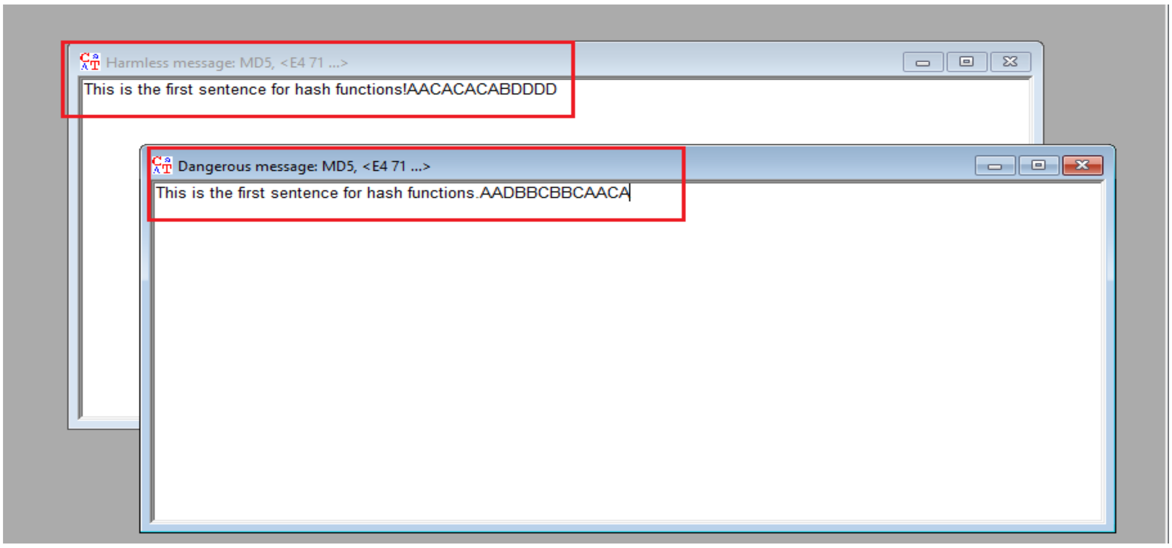
Fig. 14. Statistics of the cyber-attack



Statistics of the attack

```
Partial MD5-Collision Search

Filename original: C:\Users\pesho\Desktop\pesho_text1.txt
Filename fake: C:\Users\pesho\Desktop\pesho_text2.txt

PROJECTED EFFORTS

Calculating time:  0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.03 second(s)
Steps required


COMPUTING EFFORTS

Calculating time:  0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.00 second(s)
Steps required
Hash operations performed

RunNo.    Steps until collision    Check of the collision    Total steps

  01              2,970                    2,884                5,854


TEXT MODIFICATION

13 bytes were added to the harmless message.
13 bytes were added to the dangerous message.
```

Fig. 15. Statistics of the cyber-attack


Harmless message: MD5, <E4 71 ...>
Dangerous message: MD5, <E4 71 ...>

This is the first sentence for hash functions!AACACACABDDDD

This is the first sentence for hash functions.AADBBCBBCAACA

Fig. 16. The adding of 13 bytes to the both text messages

5. Conclusion

This scientific paper explores the practical applications of MD5, SHA-1, and SHA-256 [1,4,5,6,7,15,16,21,22,33,36] hash functions through five software tools, each addressing diverse needs and target audiences. PowerShell is the most suitable for automation in IT settings, whereas BullZip MD5 Calculator offers a simple method for basic file verification using MD5 [19,23,26,27,34]. The software tool HashCalc provides flexibility by supporting various hash functions and the software program CrypTool [2,8,9,11,19,20,24,28,29,32] is distinguished as an educational resource that is perfect for academic and research-focused analysis of hash functions. Collectively, these programs demonstrate practical significance and flexibility of the hash algorithms MD5, SHA-1, and SHA-256 [1,31,35] across different cryptographic and integrity-checking software based applications. In this context, the highly advanced laboratories at the Faculty of Technical Sciences at Konstantin Preslavsky University of Shumen provide significant opportunities for students studying [4,5] "Communication and Information Systems", "Computer Technologies in Automated Manufacturing" and "Signal Security Systems and Technologies" in order to acquire substantial theoretical and practical experience in the application of hash functions MD5, SHA-1, and SHA-256 using various software tools to verify the integrity of files [4,5].

References:

[1] Abu-El Humos, A., Alhalabi, B., Hamzal, M. K., Shufro, E., & Awada, W. (2005, November). Remote labs environments (RLE): A constructivist online experimentation in science, engineering, and information technology. In 31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005. (pp. 6-pp). IEEE.

[2] A. Z. Mahfud, M. T. Sabila, N. A. Wibowo, A. Rhamdhan and D. F. Priambodo, "A Systematic Literature Review on Operating System Security: Distribution and Issues," 2023 3rd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS), Yogyakarta, Indonesia, 2023, pp. 70-75, https://doi:10.1109/ICE3IS59323.2023.10335475, https://ieeexplore.ieee.org/abstract/document/10335475.

[3] Bedzhev, B., Trıfonov, T., & Nıkolov, N. (2010). A multicore computer system for design of stream ciphers based on random feedback shift registers. İstanbul Aydın Üniversitesi Dergisi, Turkey, 2(7), 1-15., https://dergipark.org.tr/en/download/article-file/319309. [Last accessed on 25 September 2024]

[4] Boyanov, P., Using modified sniffer scripts, implementing linear algorithms for detection of network port scan attacks in Linux based operating systems. A refereed Journal Scientific and Applied Research, Konstantin Preslavsky University Press, Vol. 24, Shumen, 2023, ISSN 1314-6289 (Print), ISSN 2815-4622 (Online), pp. 78-88.

[5] Boyanov, P., Investigating the network traffic using the command-line packets sniffer Tcpdump in Kali Linux. A refereed Journal Scientific and Applied Research, Konstantin Preslavsky University Press, Vol. 25, Shumen, 2023, ISSN 1314-6289 (Print), ISSN 2815-4622 (Online), pp. 31-44, doi: https://doi.org/10.46687/jsar.v25i1.378.

[6] Chaudhary, S., Amgai, R., Gupta, S. D., Iftekhar, N., Zafar, S., & Mahto, A. K. (2022). Comparative Study of Static and Hybrid Analysis Using Machine Learning and Artificial Intelligence in Smart Cities. In IoT for Sustainable Smart Cities and Society (pp. 195-226). Cham: Springer International Publishing.

[7] Dehghantanha, A., & Dargahi, T. (2017). Residual Cloud Forensics: CloudMe and 360Yunpan as case studies. In Contemporary Digital Forensic Investigations of Cloud and Mobile Applications (pp. 247-283). Syngress.

[8] Esslinger, B. (2008). CrypTool. Available via www.cryptool.de. [Last accessed on 19 September 2024]

[9] Esslinger, B. (2024). Learning and Experiencing Cryptography with CrypTool and SageMath. Artech House, ISBN: 978-1-68569-017-5.

[10] Gueorguiev N.L., Nesterov K.N., Minev S., An approach to information exchange management in multimodule multi-position security systems. International Scientific Journal "Security & Future", Vol. 6, Issue 1, pp: 28-

31, STUME, 2022, WEB ISSN 2535-082X; PRINT ISSN 2535-0668, https://stumejournals.com/journals/confsec/2022/1/28.full.pdf.

[11] Hick, S., Esslinger, B., & Wacker, A. (2012). Reducing the complexity of understanding cryptology using CrypTool. In 10th International Conference on Education and Information Systems, Technologies and Applications (EISTA 2012), Orlando, Florida, USA.

[12] Iliev, R., K. Ignatova. Cloud technologies for building data center system for defense and security. T. Tagarev et al. (eds.), Digital Transformation, Cyber Security and Resilience of Modern Societies, Studies in Big Data 84, ISBN 978-3-030-65721-5, Springer 2020, pp.13-24,https://doi.org/10.1007/978-3-030-65722-2.

[13] Iliev, R., Kochankov, M., A Generalized Net Model of Command and Control System. In Proceedings of the 15th International Scientific and Practical Conference, Environment. Technology. Resources. Rezekne, Latvia, Volume II, pp. 127-131, Print ISSN 1691-5402, Online ISSN 2256-070X, https://doi.org/10.17770/etr2024vol2.8035.

[14] Ivanov, I., & Aleksandrova, K. (2024, June). Design and Implementation of Software-Defined Doppler Radar. In Proceedings of the 15th International Scientific and Practical Conference, Environment. Technology. Resources. Rezekne, Latvia, Volume III, pp. 105-108, Print ISSN 1691-5402, Online ISSN 2256-070X, https://doi.org/10.17770/etr2024vol3.8159.

[15] Kao, C. N., Si, S., Huang, N. F., Liao, I. J., Liu, R. T., & Hung, H. W. (2015, April). Fast proxyless stream-based anti-virus for Network Function Virtualization. In Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft) (pp. 1-5). IEEE.

[16] Kawala, J. (2023). Encrypto: Technical Report (Doctoral dissertation, Dublin, National College of Ireland).

[17] Kochankov, M., & Iliev, R. (2024, June). A Generalized Net Model for Accessing Information Resources in a Secure Environment. In Proceedings of the 15th International Scientific and Practical Conference, Environment. Technology. Resources. Rezekne, Latvia, Volume II, pp. 175-178, Print ISSN 1691-5402, Online ISSN 2256-070X, https://doi.org/10.17770/etr2024vol2.8034.

[18] Kolev, Al., Nikolova, P., Instrumental Equipment for Cyberattack Prevention. Information & Security: An International Journal 47, no. 3 (2020):285-299. https://doi.org/10.11610/isij.4720.

[19] Kopal, N. (2018, June). Solving Classical Ciphers with CrypTool 2. In HistoCrypt (pp. 149-010).

[20] Kopal, N., & Esslinger, B. (2022, June). New Ciphers and Cryptanalysis Components in CrypTool 2. In International Conference on Historical Cryptology (pp. 127-136).

[21] Li, S., Zhang, K., Zhang, R., Zhu, M., Luo, H., & Wu, S. (2021, July). Random Parameter Normalization Technique for Mimic Defense Based on Multi-queue Architecture. In International Conference on Artificial Intelligence and Security (pp. 321-331). Cham: Springer International Publishing.

[22] Lieber, P. A. (2011). FPG Communication Framework for Communication, Debugging, Testing, and Rapid Prototyping (Master's thesis, Brigham Young University).

[23] Llovet Ureña, M. (2015). Network layer security and secret key authentication (Bachelor's thesis, Universitat Politècnica de Catalunya).

[24] Maeref, M., & Algali, F. (2015, January). An empirical evaluation of Cryptool in teaching computer security. In Proceedings of the International Conference on Computer Science, Engineering and Applications (pp. 93-100).

[25] Mirtcheva-Ivanova, Daniela, Application of electronic platforms to increase the knowledge of learners. In Proceedings of the 15th International Scientific and Practical Conference, Environment. Technology. Resources. Rezekne, Latvia, Volume II, pp. 448-452, Print ISSN 1691-5402, Online ISSN 2256-070X, https://doi.org/10.17770/etr2024vol2.8090.

[26] Mirtcheva-Ivanova, D., Application of Artificial Intelligence in E-Learning. In Proceedings of the 15th International Scientific and Practical Conference, Environment. Technology. Resources. Rezekne, Latvia, Volume II, pp. 208-211, Print ISSN 1691-5402, Online ISSN 2256-070X, https://doi.org/10.17770/etr2024vol2.8053.

[27] Moreb, M. (2022). Impact of Device Jailbreaking or Rooting on User Data Integrity in Mobile Forensics. In Practical Forensic Analysis of Artifacts on iOS and Android Devices: Investigating Complex Mobile Devices (pp. 227-258). Berkeley, CA: Apress.

[28] Nurdin, A. A., & Djuniadi, D. (2022). Securing audio chat with cryptool-based twofish algorithm. Journal of Soft Computing Exploration, 3(1), 37-43.

[29] Onete, C. (2008). Visualisation of Modern Key Exchange Schemes for more than two Parties in CrypTool and their Security Analysis.

[30] Pavlov, G., Kolev. Al., A place of GIS technologies in information Systems for crisis prevention, 6th International Conference on Application of Information and Communication Technology and Statistics In Economy and Education (ICAICTSEE – 2016), December 2-3rd, 2016, UNWE, Sofia, Bulgaria, ISSN 2367-7635 (print), ISSN 2367-7643 (online), pp. 452-457.

[31] Qureshi, M. A., Ahmed, S., Mehmood, A., Shaheen, R., & Dildar, M. S. (2024). Vulnerability assessment of operating systems in healthcare: exploitation implications techniques and security. Health Sciences Journal, 2(2), 104-111, ISSN (Online): 2959-2259, ISSN (Print): 2959-2240, https://doi.org/10.59365/hsj.2(2).2024.98.

[32] Salmi, G. N., & Siagian, F. (2022). Implementation of the data encryption using caesar cipher and vernam cipher methods based on CrypTool2. Journal of Soft Computing Exploration, 3(2), 99-104.

[33] Sobieraj, S. C. (2008). Mobile device forensics case file integrity verification. Master of Science thesis, Purdue University, West Lafayette, Indiana.

[34] Sudyana, D., Putra, R. T., & Soni, S. (2019). Digital Forensics Investigation on Proxmox Server Virtualization Using SNI 27037: 2014. Sinkron: jurnal dan penelitian teknik informatika, 3(2), 67-72.

[35] Suhaili, S., Julai, N., Sapawi, R., & Rajaee, N. (2024). Towards Maximising Hardware Resources and Design Efficiency via High-Speed Implementation of HMAC based on SHA-256 Design. Pertanika Journal of Science & Technology, 32(1).

[36] Tuli R. Analyzing Network performance parameters using wireshark. arXiv preprint arXiv:2302.03267, 2023 Feb 7, International Journal of Network Security & Its Applications (IJNSA), 2023.

[37] Trifonov T., 2019, Modeling and Calculation of Passive Audio Crossovers, Annual of Konstantin Preslavsky University of Shumen, Vol IX E Technical Sciences, ISSN 1311-834X, pp. 182-189.

[38] Trifonov, T., Performance analysis of a mobile computer equipped with solid state disk. Annual of Konstantin Preslavsky University of Shumen, Shumen, Konstantin Preslavsky University Press, ISSN 1311-834X, Vol. IV E, 2014, pp. 27–42.