CS EE World
https://cseeworld.wixsite.com/home
29/34 (A)
May 2022

Donor Info:
Discord: <3#7654

Computer Science Extended Essay

TOPIC: hashing algorithms for secure data transfer over a network.

Research Question: Which hashing algorithm out of MD5 and SHA -1 is the best in validating data transfer using SFTP (simple file transfer protocol) over a network that implements SSH protocol in terms of speed of data transfer and collision resistance?

Word count: 3144

Table of Contents

Table of figures 2
Introduction 3
Background of the topic 3
What are hashing algorithms? 3
Terminology 4
Statistics 7
Primary Research 11
Methodology 11
Data 12
Analysis of the data 14
Conclusion 16
Evaluation 16
Limitations 16
Unanswered questions 16
Bibliography 17
Appendix 1 18

Table of figures

Figure 1 Collision of MD5 7
Figure 2 Collision of SHA-1 8
Figure 3 Bits hashed per second for different algorithms 10
Figure 4 transfer rates for different file sizes 14
Figure 5 Differences of transfer rates for different file sizes 14
Figure 6 Transfer of 1 GB file 18
Figure 7 Transfer of 2 GB file 18
Figure 8 Transfer of 3 GB file 19
Figure 9 Transfer of 5 GB file 19
Figure 10 Transfer of 100 MB file 20
Figure 11 Transfer of 300 MB file 20
Figure 12 Transfer of 500 MB file 21

Introduction

By the early 1990s, secure transaction of data over the internet was a major concern. SSH (Secure Shell Protocol), was a major contribution for the internet community, especially for web developers. SSH guaranteed privacy for the users over the internet, allowing to transmit the data over networks without the data being tampered by unauthorized individuals. how the validity of the data was ensured? Validity is guaranteed using various techniques, but the core lies in hashing algorithms. SSH-2.0 supports various hashing algorithms[1], including MD5 and SHA 1. Various studies were conducted in comparison of different hashing algorithms, which were implemented in authentication systems like Simple-O[2] , showing different advantages of the hashing algorithms in different situations over a network. As different algorithms have different methods of hashing, factor such as speed of data transfer is affected. Although the algorithms were studied deeply, the cybersecurity scene is always changing, therefore making outdated research inaccurate. This extended essay will compare the implications of SHA 1 and MD5 on SSH, therefore demonstrating which hashing algorithm is more adequate for the implemented protocol. This extended essay will first discuss about hashing algorithms and their implementations, leading into the primary research, where data would be analyzed, therefore allowing to conclude which algorithm is viable for the SSH protocol.

Background of the topic

What are hashing algorithms?

During the early days of the internet, checking authenticity of information was a major security problem. To partially resolve this problem, hash functions were implemented and authentication techniques were developed, based on multiple hashing algorithms. Hashing algorithms compress input message of practically any length, into a “fingerprint” of a fixed length, without provision of any secret parameter. Each output message has a unique fixed length, which is determined by the algorithm used.

The principal security properties[3] of hash functions are:

1.Pre-image resistance: It’s “computationally infeasible” to find the input message.

2. Second preimage resistance: It’s impossible to find an input, which gives a same output as any specified input.

3.Colllision resistance: It’s hard to find 2 different inputs which give the same output.

[1] SSH.COM Tectia SSH Client/Server - Secure File Transfer and Remote Access, SSH.COM, www.ssh.com/products/tectia-ssh/.

[2] A. A. Putri Ratna, P. Dewi Purnamasari, A. Shaugi and M. Salman, "Analysis and comparison of MD5 and SHA-1 algorithm implementation in Simple-O authentication based security system," 2013 International Conference on QiR.

[3] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography, chapter Hash Functions and Data Integrity, pages 323–324. The CRC Press series on discrete mathematics and its applications. CRC Press, 1997.

Modern hashes are based on an iterative construction proposed by Merkle[4]/Damgard[5]. Hashes which obey properties 1 and 2 are called “one-way” hash functions, whereas hashes which obey all the properties are called “collision resistant” hash functions. Hash functions have a widespread use, starting in storage by the implementation of hash tables, to cryptocurrencies which use hash functions to control the network.

Terminology

Over the years, the exponential rise of computational power[6], leads to an increase of terminology as new technology was developed. Terminologies which are frequently used in this work are explained below.

MD5: A message digest developed by R. Rivest, in the year 1991. It’s a simple algorithm, allowing it to effectively run on 32-bit machines. It’s generally used to get a checksum of a file to verify data integrity, but its implementation is wide such as partition keys in a database.

The working of the algorithm[7]:

1. The input message bits are padded to be 64 bits short of a multiple of 512 bits. It’s done by appending 1, and then 0 until the desired length is reached.
2. A 64-bit representation of the length of the original message is appended to the current message.
3. The MD5 buffer is initialized, Containing 4 32-bit registers. The values initialized are hexadecimal.
4. The message is processed in 16-Word blocks. It is done by initializing conditional functions and performing mathematical operations
5. The final message can be combined using different variables, which values were found during the previous step.

SHA-1: a hashing algorithm designed by the NSA (natural security agency), and was first published in 1993. This hashing algorithm had a myriad of applications, which included fingerprinting for data verification and digital certification. It’s based on a similar structure, which was implemented in MD5.

Working of the algorithm:

1. Input message is broken into 512-bit blocks.
2. If the message isn’t divisible by 512, the message is padded to be divisible by 512.
3. The SHA 1 buffer is initialized, containing five 32-bit registers. The values initialized are hexadecimal.
4. The message is processed in a 160-bit state. Each 32-bit block goes through 80 operations formed upon non-linear functions, left rotations and modular addition.

[4] Ralph C. Merkle. One way hash functions and DES. Lecture Notes in Computer Science, 435:428–446, 1990.

[5] I.B. Damgard. A design principle for hash functions. Lecture Notes in Computer Science, 435:416–427, 1990.

[6] Moore, Gordon E. (1965-04-19). "Cramming more components onto integrated circuits" (PDF). intel.com. Electronics Magazine. Retrieved July 14, 2021.

[7] Rivest, R. (April 1992). "MD5 Algorithm Description Blocks". The MD5 Message-Digest Algorithm. IETF. p. 5. sec. 3.4. doi:10.17487/RFC1321. RFC 1321. Retrieved 14 July 2021.

5. The processed chunks are combined into a 160-bit string, which is the output value of the hash.

SSH: SSH is a software package that enables secure system administration and file transfers over insecure networks.[8] The protocol uses a client- server model between the SSH client and the SSH server. It was developed by Tatu Ylönen, as a response to a packet-sniffing attack at his university.

Working of the protocol:[9]

1. The client establishes a connection with the server.
2. Server sends its public RSA host key and another RSA public key, which is then compared to the host keys stored in a database on the client's machine.
3. Using a cryptographic number generator, the client generates a 256-bit random number. The client chooses a cryptographic algorithm supported by the server. The client encrypts the number, which would be used as the session key, using the host key and the server key and sends the key to the server.
4. The server decrypts the session key, and sends an encrypted confirmation to the client. The connection between the server and the client is encrypted by the session key.
5. The client undergoes authentication from the server. It can be done using a username and password, or using an RSA-based host authentication.
6. Client sends request to set up the type of session needed, such as TCP-IP forwarding, X11 forwarding, etc.
7. The client and the server exchange packet asynchronously. To end the connection, the client sends a termination message to the server, which is replied by a termination message from the server. The client closes the connection.

SFTP: Secure file transfer protocol, which runs over SSH, therefore inheriting all the security and authentication features of SSH.[10] The protocol was developed by the Internet Engineering Task Force in the year 1997. The protocol allows the user to access, browse, transfer and manage data on a remote machine.

HMAC:[11] keyed-hash message authentication code allows us to validate data which has been transmitted between devices over a network. The algorithm allows us to validate data without significantly loosing performance, and has cryptographic functions which can run on most devices currently implemented. The algorithm also allows us to use different hashing algorithms, in case security vulnerabilities are found in outdated algorithms, and new hashing functions become available.

[8] "SSH Secure Shell Home Page, Maintained by SSH Protocol Inventor Tatu Ylonen. SSH Clients, Servers, Tutorials, How-Tos." SSH Secure Shell Home Page, Maintained by SSH Protocol Inventor Tatu Ylonen. SSH Clients, Servers, Tutorials, How-Tos., www.ssh.com/academy/ssh.

[9] Ylonen, Tatu. "SSH–secure login connections over the Internet." Proceedings of the 6th USENIX Security Symposium. Vol. 37. 1996.

[10] "SFTP Protocol, Clients, Servers Etc.. Page by the Original Author of SFTP." SFTP Protocol, Clients, Servers Etc. Page by the Original Author of SFTP., https://www.ssh.com/academy/ssh/sftp.

[11] Krawczyk, Hugo, Mihir Bellare, and Ran Canetti. "HMAC: Keyed-hashing for message authentication." (1997).

The code is calculated using this function:

H(K XOR opad, H(K XOR ipad, data))

Explanation of the variables:

- H: the hashing algorithm
- K: the authentication key. The key's length is limited by the byte length of the outputs from the hashing algorithms (16 for MD5 and 20 for SHA-1)
- opad : the byte 0x5C repeated 64 times
- ipad : the byte 0x36 repeated 64 times

Working of the function:

1. Zeroes are appended to K, to ensure that it matches the byte length of the cryptographic function implemented.
2. XOR is computed between the value from step 1 and ipad.
3. The data is appended to the value from step 2.
4. The cryptographic function is applied to the value generated from step 3.
5. XOR is computed between the value from step 1 and opad
6. The result from step 4 is appended to the result from step 5
7. Cryptographic function H is applied to the value generated in step 6 to generate the output

Statistics

Several previous researches compared SHA 1 and MD5 in different environments. The studies are shown below. The hashes were compared in terms of collisions, hash rate and effectiveness in different authentication systems.

Collisions: two distinct points in the domain of a hash function that hash to the same range point.[12] This has several implications, as data can be altered without changing the output hash.

MD5[13]: In 1996, Dobbertin[14] was able to demonstrate that collisions in MD5 were theoretically possible. The requirement was that initialization vector should be chosen, but in practice the initialization vector is set, therefore not posing a severe threat towards the hash. In the year 2004, wang et al. discovered collisions with the standard initialization vectors. A MD5 collision is demonstrated in table 1.

Message 1	1st block	02DD31D1	C4EEE6C5	069A3D69	5CF9AF98	87B5CA2F	AB7E4612
		3E580440	897FFBB8	0634AD55	02B3F409	8388E483	5A417125
		E8255108	9FC9CDF7	F2BD1DD9	5B3C3780		
	2st block	D11D0B96	9C7B41DC	F497D8E4	D555655A	C79A7335	0CFDEBF0
		66F12930	8FB109D1	797F2775	EB5CD530	BAADE822	5C15CC79
		DDCB74ED	6DD3C55F	D80A9BB1	E3A7CC35		
Message 2	1st block	02DD31D1	C4EEE6C5	069A3D69	5CF9AF98	87B5CA2F	AB7E4612
		3E580440	897FFBB8	0634AD55	02B3F409	8388E483	5A417125
		E8255108	9FC9CDF7	72BD1DD9	5B3C3780		
	2st block	D11D0B96	9C7B41DC	F497D8E4	D555655A	479A7335	0CFDEBF0
		66F12930	8FB109D1	797F2775	EB5CD530	BAADE822	5C154C79
		DDCB74ED	6DD3C55F	580A9BB1	E3A7CC35		
MD5 Hash		8D5E7019	6324C015	715D6B58	61804E08		

Figure 1 Collision of MD5

[12] Rogaway, Phillip, and Thomas Shrimpton. "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance." International workshop on fast software encryption. Springer, Berlin, Heidelberg, 2004.

[13] Thompson, Eric. "MD5 collisions and the impact on computer forensics." Digital investigation 2.1 (2005): 36-40.

[14] Dobbertin, Hans. "Cryptanalysis of MD5 compress." rump session of Eurocrypt 96 (1996): 71-82.

SHA 1: In 1997, Wang[15] demonstrated the first attacks on the SHA archetype. Presently, collisions can be found if the initialization vector is chosen. Once near- collisions are found, two block collisions can be constructed. Even through collisions are possible, the time complexity is very high. For example, 10 days of calculation by 64 GPUs is required to find a viable collision, making $2^{57.5}$ calls to the SHA hash function. [16] Collisions of the sha-1 function is demonstrated in table 2.

Message 1	Input	4e	a9	62	69	7c	87	6e	26	74	d1	07	f0	fe	c6	79	84	14	f5	bf	45
	1st Block			7f	46	dc	93	a6	b6	7e	01	3b	02	9a	aa	1d	b2	56	0b		
				45	ca	67	d6	88	c7	f8	4b	8c	4c	79	1f	e0	2b	3d	f6		
				14	f8	6d	b1	69	09	01	c5	6b	45	c1	53	0a	fe	df	b7		
				60	38	e9	72	72	2f	e7	ad	72	8f	0e	49	04	e0	46	c2		
	output	8d	64	d6	17	ff	ed	53	52	eb	c8	59	15	5e	c7	eb	34	f3	8a	5a	7b
	2nd block			30	57	0f	e9	d4	13	98	ab	e1	2e	f5	bc	94	2b	e3	35		
				42	a4	80	2d	98	b5	d7	0f	2a	33	2e	c3	7f	ac	35	14		
				e7	4d	dc	0f	2c	c1	a8	74	cd	0c	78	30	5a	21	56	64		
				61	30	97	89	60	6b	d0	bf	3f	98	cd	a8	04	46	29	a1		
	output	1e	ac	b2	5e	d5	97	0d	10	f1	73	69	63	57	71	bc	3a	17	b4	8a	c5
Message 2	input	4e	a9	62	69	7c	87	6e	26	74	d1	07	f0	fe	c6	79	84	14	f5	bf	45
	1st Block			73	46	dc	91	66	b6	7e	11	8f	02	9a	b6	21	b2	56	0f		
				f9	ca	67	cc	a8	c7	f8	5b	a8	4c	79	03	0c	2b	3d	e2		
				18	f8	6d	b3	a9	09	01	d5	df	45	c1	4f	26	fe	df	b3		
				dc	38	e9	6a	c2	2f	e7	bd	72	8f	0e	45	bc	e0	46	d2		
	output	8d	64	c8	21	ff	ed	52	e2	eb	c8	59	15	5e	c7	eb	36	73	8a	5a	7b
	2nd block			3c	57	0f	eb	14	13	98	bb	55	2e	f5	a0	a8	2b	e3	31		
				fe	a4	80	37	b8	b5	d7	1f	0e	33	2e	df	93	ac	35	00		
				eb	4d	dc	0d	ec	c1	a8	64	79	0c	78	2c	76	21	56	60		
				dd	30	97	91	d0	6b	d0	af	3f	98	cd	a4	bc	46	29	b1		
	output	1e	ac	b2	5e	d5	97	0d	10	f1	73	69	63	57	71	bc	3a	17	b4	8a	c5

Figure 2 Collision of SHA-1

Several other comparative studies were conducted. As shown below, a comparative study between the hashing algorithms demonstrating difference in hashing rates and bits per second.[17] Different strings were used.

[15] X. Y. Wang. The Collision attack on SHA-0. In Chinese, to appear on www.infosec.edu.cn, 1997.

[16] Stevens, M., Lenstra, A., Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007).

[17] Long, Sihan. "A Comparative Analysis of the Application of Hashing Encryption Algorithms for MD5, SHA-1, and SHA-512." Journal of Physics: Conference Series. Vol. 1314. No. 1. IOP Publishing, 2019.

MD5:

Character string	“abc”
Hash Value	900150983cd24fb0d6963f7d28e17f72
Average Running Time	3.08556 ms
Bits per second	0.97227 bit/ms

Character string	“followingabcdefghijklmnopqrstuvwxyz”
Hash Value	21bb14b92f9d6f6205868eedeefcd4da
Average Running Time	5.37620 ms
Bits per second	6.51017 bit/ms

Character string	“qwertyuiopasdfghjklzxcvbnmpoiuytrewqasdfghjklmnbvcxzmnbvcxzasdfghjklpoiuytrewq”
Hash Value	779290ab2fb7ebc2aa91805df8559b39
Average Running Time	7.75992 ms
Bits per second	10.05168 bit/ms

SHA-1:

Character string	“abc”
Hash Value	a9993e364706816aba3e25717850c26c9cd0d89d
Average Running Time	8.87031 ms
Bits per second	0.33821 bit/ms

Character string	“followingabcdefghijklmnopqrstuvwxyz”
Hash Value	916e4febe8aae0e2438846d26db0dc2333f90ba6
Average Running Time	9.22622 ms
Bits per second	3.79353 bit/ms

Character string	“qwertyuiopasdfghjklzxcvbnmpoiuytrewqasdfghjklmnbvcxzmnbvcxzasdfghjklpoiuytrewq”
Hash Value	c200f0c3a9a671388a65c8f7546376cafa575fac
Average Running Time	15.60673 ms
Bits per second	4.99784 bit/ms

As shows above, there is a large difference between the running time between SHA and MD5. The number of bits per second for MD5 is larger in comparison to SHA 1, showing the capability of MD5 in processing large strings without the loss of efficiency.

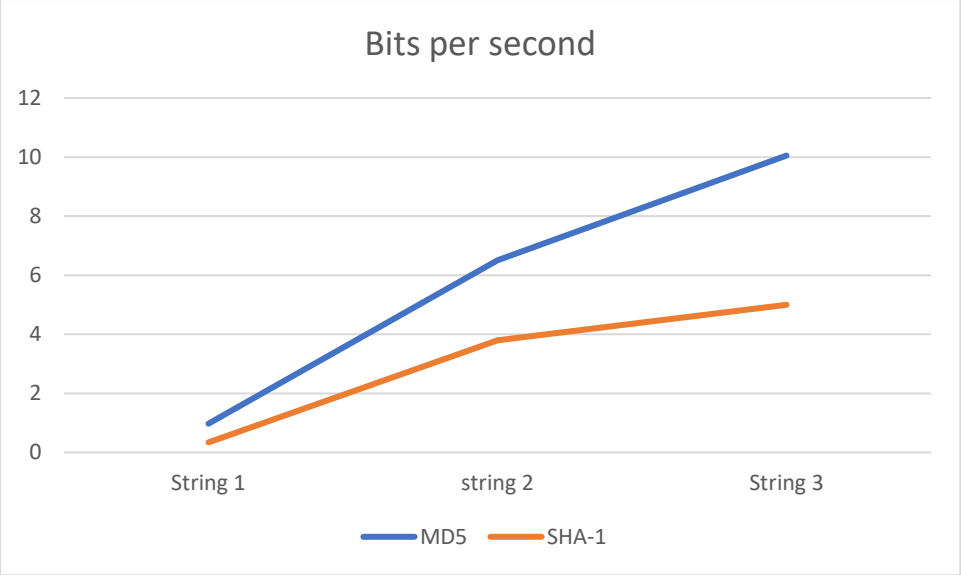

Figure 3 Bits hashed per second for different algorithms

Time complexities:

MD5:

$\sum = T(n) = \theta(n + n\times(16 + 16\times9\times2 + 16\times7 + 16\times8) + 4 + 3) = \theta(N)$

SHA-1:

$\sum = T(n) = \theta(n + n\times(16 + 64\times4 + 20\times15 + 20\times13\times2 + 20\times16) + 5 + 4) = \theta(N)$

The time complexities of the algorithms are the same.

Comparison of the algorithms:

Algorithm	Output size in characters	Time Complexity	Running Time(relatively)	Bits per hash	Block Size	Security
MD5	32	$\theta(N)$	Fast	High	512	Weak
SHA-1	40	$\theta(N)$	Slow	Low	512	Strong

Primary Research

Methodology

To measure the rates of data transfer, files were transferred over sftp protocol from an Ubuntu 21.10 to a windows server 2016. The operating systems were hosted on Virtual box 6.1. The operating systems were given the following resources:

- RAM: 10 000 MB
- Chipset: PIIX3
- Processors: 4 @ 0.35 (Intel® core™ i7-7700HQ CPU)
- Video memory: 128 mb
- Network adapter: Intel PRO/1000 MT Desktop (82540EM)
- Drive: 40 GB Virtual Hard disks, allocated on an SSD.

The operating systems were connected using a bridged adapter, to provide the most optimal networking conditions for the protocols. The operating systems used the following clients for the SSH connections: V8.6.0.0p1-Beta OpenSSH for the windows server and OpenSSH 8.6 for the ubuntu.

Additionally, I used the following file sizes to determine the effect of transfer rate in respect to the file size: 5GB, 3GB, 2GB, 1GB, 500mb, 300mb and 100mb. The dummy files were created on the windows server using the following command:

```
fsutil file createnew <filename> <length>
```

The size of the file was specified in bytes.

To establish the SFTP connection between the devices, the following commands were executed on the ubuntu terminal

```
1. sftp -oMACs=hmac- <sha1 or md5>  ADMINISTRATOR@192.168.1.107
2. cd /C:/OHNO
3. get <filename>
4. exit
```

Explanation of the procedure:

1. calling the sftp program in terminal. Setting the HMAC method as SHA1 or MD5. Specifying the username and host ip address, for establishing the ssh connection.
2. Navigating to the folder on the remote machine, to find the files required to be transferred.
3. Telling the SFTP protocol to copy the file from the remove machine into the directory from which the terminal was active.
4. Terminating the SFTP connection between the devices.

The following commands were used several times to acquire the primary data, transferring files. The average rate of the data transfer was shown at the end of each transfer.

Data

The data used to test the transfer rates are dummy files. Files of different sizes are used to investigate the potential effectiveness of a hashing algorithm for a particular file size. The files were transferred over a virtual network, providing perfect network conditions, minimizing packet routing and latency issues. To account for potential packet routing and buffer issues, the file transfer was executed five times. The data represented in megabytes per second, which can be found by dividing the data transferred by the amount of time taken to transfer the data. This method won't be required as the average rate of transfer would be shown by the SFTP client (Appendix 1). The data demonstrated below displays the transfer rates obtained for each algorithm in respect to each file size.

5 GIGABYTES	
MD5(MB/s)	SHA – 1(MB/s)
91.5	79.1
76.1	74.1
85.7	76.0
71.2	84.2
70.2	75.7

3 GIGABYTES	
MD5 (MB/s)	SHA – 1(MB/s)
76.4	80.8
62.7	77.9
75.5	74.1
72.4	80.1
71.8	72.7

2 GIGABYTES	
MD 5 (MB/s)	SHA-1(MB/s)
66.6	75.1
78.6	69.1
86.1	70.2
71.8	74.1
82.5	71.6

1 GIGABYTE	
MD 5(MB/s)	SHA -1(MB/s)
74.3	65.3
70.5	66.00
79.8	67.4
72.2	66.2
68.0	64.9

500 MEGABYTES	
MD 5 (MB/s)	SHA 1(MB/s)
86.4	64.3
80.3	65.2
77.8	70.4
91.0	73.9
70.9	71.5

300 MEGABYTES	
MD 5 (MB/s)	SHA 1(MB/s)
79.6	73.7
82.8	72.5
70.2	69.2
68.0	72.3
81.4	86.7

100 MEGABYTES	
MD 5 (MB/s)	SHA 1(MB/s)
91.2	64.5
90.9	65.1
75.1	70.1
68.9	66.7
83.4	66.5

As we can see from the previous data, the transfer rates varied due to several factors. Finding an average takes the variance into account, showing the overall potential transfer rates. The average was found by adding all the data transfer values for a particular algorithm and file size and divided by 5. The table shown below displays the averages found:

File size	MD5(MB/s)	SHA -1(MB/s)
5 GIGABYTES	78.94	72.82
3 GIGABYES	71.8	77.12
2 GIGABYES	77.12	72.02
1 GIGABYTE	72.96	65.96
500 MEGABYTES	81.28	69.06
300 MEGABYTES	76.4	75.04
100 MEGABYTES	81.9	66.58

Analysis of the data

As demonstrated in the previous section, the data shows various trends. The trend influenced towards MD5, showing a slightly different trends for different situations.

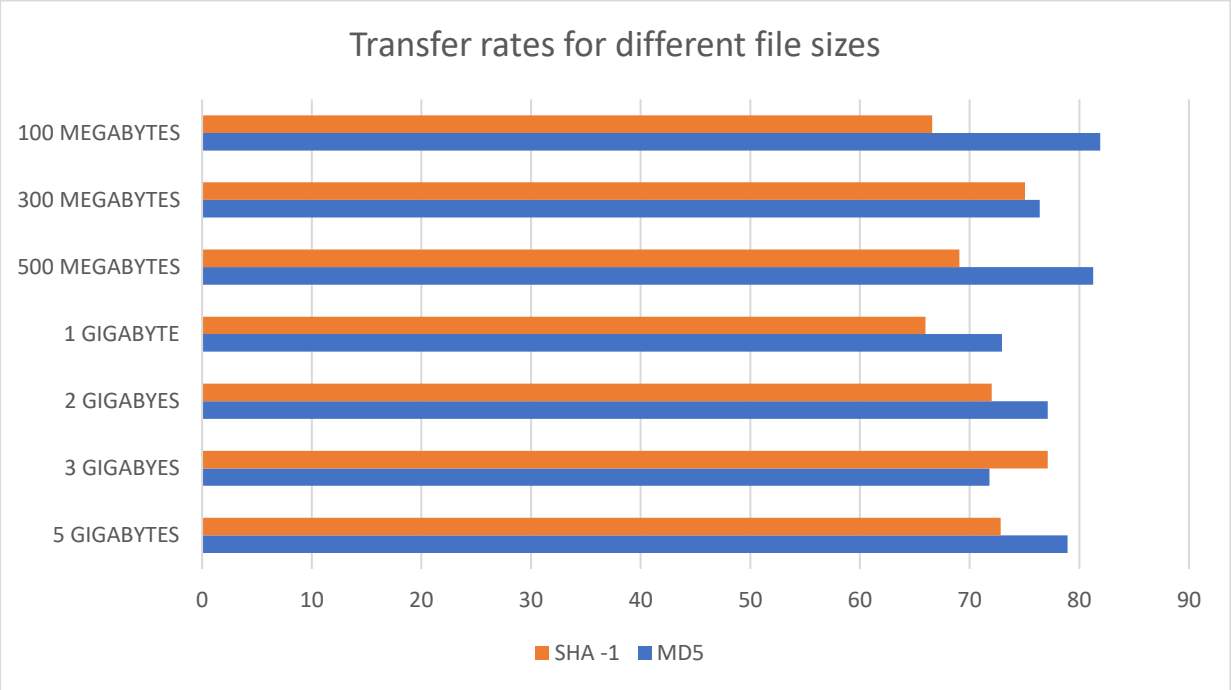

Figure 4 transfer rates for different file sizes

Shown above is a graphical representation for the averages of the data collected. The data transfer rates are represented in megabytes per second. As seen from the graph, the data transfer rates vary for different file sizes. Although MD5 provides faster data transfer for most file sizes, SHA-1 has a slight edge for 3 gigabytes. To give a better insight into the differences, a graph can be shown.

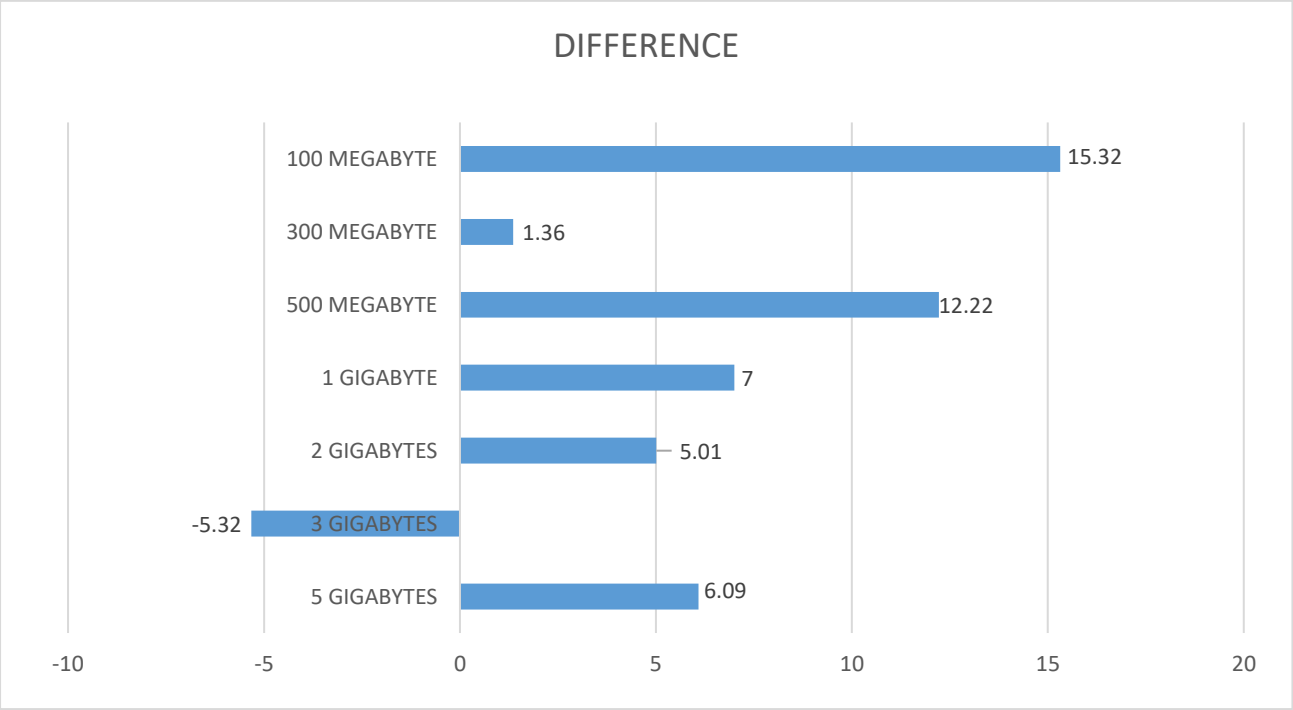

Figure 5 Differences of transfer rates for different file sizes

The difference was calculated using the Formula:

Difference = Average rate of transfer for MD5 – Average rate of transfer for SHA 1

The positive difference demonstrates the higher transfer rates for MD5. The highest difference demonstrated by MD5 was for 100 MB. This demonstrates the ability of the algorithm to churn small data blocks faster, due to the difference in buffer sizes.

At 300 megabytes and 3 gigabytes, sha 1 performs considerably better. We can conclude that SHA 1 has an anomalous performance increase with file sizes which have the largest place value starting with 3. Such an anomaly is caused due to the algorithms initializations vectors being able to process these numbers with a higher efficiency.

The performance gap has a noticeable decrease for larger file sizes. Even through the SHA 1 performs well for 3GB, the performance metrics for large files such as 2 GB and 5 GB are clearly in favor of MD5. As we saw previously in the statistics portion of this work, MD5 has a higher hash rate than SHA -1, possibly leading us to a conclusion that transfer for large data files should be exponentially faster. Although the data doesn't show the trend, pointing us towards other limiting factors for the data transfer, such as the SSH protocol.

The SSH 2 protocol enforces multiplexing over a single Transmission control protocol window. To integrate the SSH protocol with minimal collisions with different protocols, the SSH receive window configuration is similar to the TCP receive window configuration. This results in an application receive window on top the TCP receive window. This effectively limited the application receive window towards the size of the TCP receive window. Although the receive window is dynamically allocated in the operating system used for this work (Ubuntu 21.10), the default window size set is 64 kb[18]. This severely impacts the performance of the algorithm in terms of data transfer of large files. At a point where large data transfers are performed, most of the buffer space is used up. At this point, most modern processors can perform cryptographic operations fast enough to keep up with the data transfer rate, having a lesser impact on the overall transfer speed. A possible solution is to increase the TCP window size, but such an action might cause over buffering issues, negatively impacting the transfer rate of the SSH algorithm.

The data transfer is further limited by the SFTP protocol. SFTP transfers the data in 32kb blocks, which are considered as requests. Since the flow control is imposed on SFTP, a limited amount of data blocks can be present in transit between the client and the server. SFTP allows the client to have 16 data requests, which in total give us a transfer window of 512 kb (16 x 32). This creates a severe bottleneck for the protocol, hampering the efficiency of MD5 in transferring large files.

[18] Canonical. Ubuntu Manpage: TCP - TCP Protocol, http://manpages.ubuntu.com/manpages/bionic/man7/tcp.7.html.

Conclusion

Evaluation

In terms of data transfer, MD5 demonstrated higher performance over different file sizes, expect for a few anomalies, which aren't viable enough for general use. These tests were conducted in perfect network conditions, to reduce impacts of factors such as packet routing, reaching the bottleneck for the algorithm. Although secondary data demonstrated that MD5 has higher executing speeds, processing more data per second, therefore having significant runtime difference with larger files. Such a correlation was not demonstrated in case of transferring large files with the implementation of MD5, demonstrating a smaller transfer difference. This was due to the limitations of the TCP transfer window, leading to limitations in the SFTP transfer window. This ensured the algorithms bottleneck was reached with the data transfer of large files.

As demonstrated in the secondary research, MD5 has certain security concerns due to the simplicity of the algorithm. It was demonstrated that collisions for MD5 are easier to find in comparison to SHA 1, due to different number of initialization buffers. This could bear various security implications, as data can be manipulated during the process of transfer, which would result in altered data getting validated, as the resulting output hash of the colliding bytes would be the same. Although such attacks are still unviable, as finding a suitable collision for MD5 would take years, rendering it unviable. Although, taking into account that computational power is increasing exponentially, future attacks on the hashing algorithm might become viable. As collisions for MD5 and SHA 1 have been proven, these hashing algorithms aren't viable for validating sensitive data over a network, as the systems based on these validation methods would be targeted if a severe vulnerability would be found in the future. Benefits of MD5 could be reaped on networks which involves data transfer of files below 1GB, giving the most optimal performance. For files larger than 1 GB, implementation of MD5 wouldn't significantly affect the data transfer rates, as the algorithm reaches its bottleneck capacity.

Limitations

This research provided perfect networking conditions for the data transfer algorithm. Such environment isn't applicable pratically over WAN, as WAN has several data transfer limitations such as packet routing.

The research provided abundant system resuources for the system to operate with. Such system resources arent always available.

The algorithm reached its bottleneck transfer capacity while transfering files over 1 GB, due to the limitations of the TCP transfer window, which affected the transfer window and the amount of transfer requests preseft for the SFTP protocol.

Unanswered questions

Although this research observed several aspects of the algorithm, but alas, some questions couldn't be answered. The limitation of the TCP transfer window severely impacted the data transfer rates for large files. With sufficient calculations, custom TCP frame sizes large enough for data transfers involving large files could be configured. How would larger TCP frame sizes affect the transfer rates of the STFP algorithm and the overall functioning of the SSH protocol? This issue could be investigated, as it would be impacting in scenarios where large files need to be transferred over a network.

Bibliography

- “SFTP Protocol, Clients, Servers Etc.. Page by the Original Author of SFTP.” SFTP Protocol, Clients, Servers Etc. Page by the Original Author of SFTP., https://www.ssh.com/academy/ssh/sftp.
- “SSH Secure Shell Home Page, Maintained by SSH Protocol Inventor Tatu Ylonen. SSH Clients, Servers, Tutorials, How-Tos.” SSH Secure Shell Home Page, Maintained by SSH Protocol Inventor Tatu Ylonen. SSH Clients, Servers, Tutorials, How-Tos., www.ssh.com/academy/ssh.
- A. Putri Ratna, P. Dewi Purnamasari, A. Shaugi and M. Salman, "Analysis and comparison of MD5 and SHA-1 algorithm implementation in Simple-O authentication based security system," 2013 International Conference on QiR.
- Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography, chapter Hash Functions and Data Integrity, pages 323–324. The CRC Press series on discrete mathematics and its applications. CRC Press, 1997.
- Canonical. Ubuntu Manpage: TCP - TCP Protocol, http://manpages.ubuntu.com/manpages/bionic/man7/tcp.7.html.
- Dobbertin, Hans. "Cryptanalysis of MD5 compress." rump session of Eurocrypt 96 (1996): 71-82.
- I.B. Damgard. A design principle for hash functions. Lecture Notes in Computer Science, 435:416–427, 1990.
- Krawczyk, Hugo, Mihir Bellare, and Ran Canetti. "HMAC: Keyed-hashing for message authentication." (1997).
- Long, Sihan. "A Comparative Analysis of the Application of Hashing Encryption Algorithms for MD5, SHA-1, and SHA-512." Journal of Physics: Conference Series. Vol. 1314. No. 1. IOP Publishing, 2019.
- Moore, Gordon E. (1965-04-19). "Cramming more components onto integrated circuits" (PDF). intel.com. Electronics Magazine. Retrieved July 14, 2021.
- Ralph C. Merkle. One way hash functions and DES. Lecture Notes in Computer Science, 435:428–446, 1990.
- Rivest, R. (April 1992). "MD5 Algorithm Description Blocks". The MD5 Message-Digest Algorithm. IETF. p. 5. sec. 3.4. doi:10.17487/RFC1321. RFC 1321. Retrieved 14 July 2021.
- Rogaway, Phillip, and Thomas Shrimpton. "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance." International workshop on fast software encryption. Springer, Berlin, Heidelberg, 2004.
- SSH.COM Tectia SSH Client/Server - Secure File Transfer and Remote Access, SSH.COM, www.ssh.com/products/tectia-ssh/.
- Stevens, M., Lenstra, A., Weger, B.: Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007).
- Thompson, Eric. "MD5 collisions and the impact on computer forensics." Digital investigation 2.1 (2005): 36-40.
- X. Y. Wang. The Collision attack on SHA-0. In Chinese, to appear onwww.infosec.edu.cn, 1997.
- Ylonen, Tatu. "SSH–secure login connections over the Internet." Proceedings of the 6th USENIX Security Symposium. Vol. 37. 1996.

Appendix 1

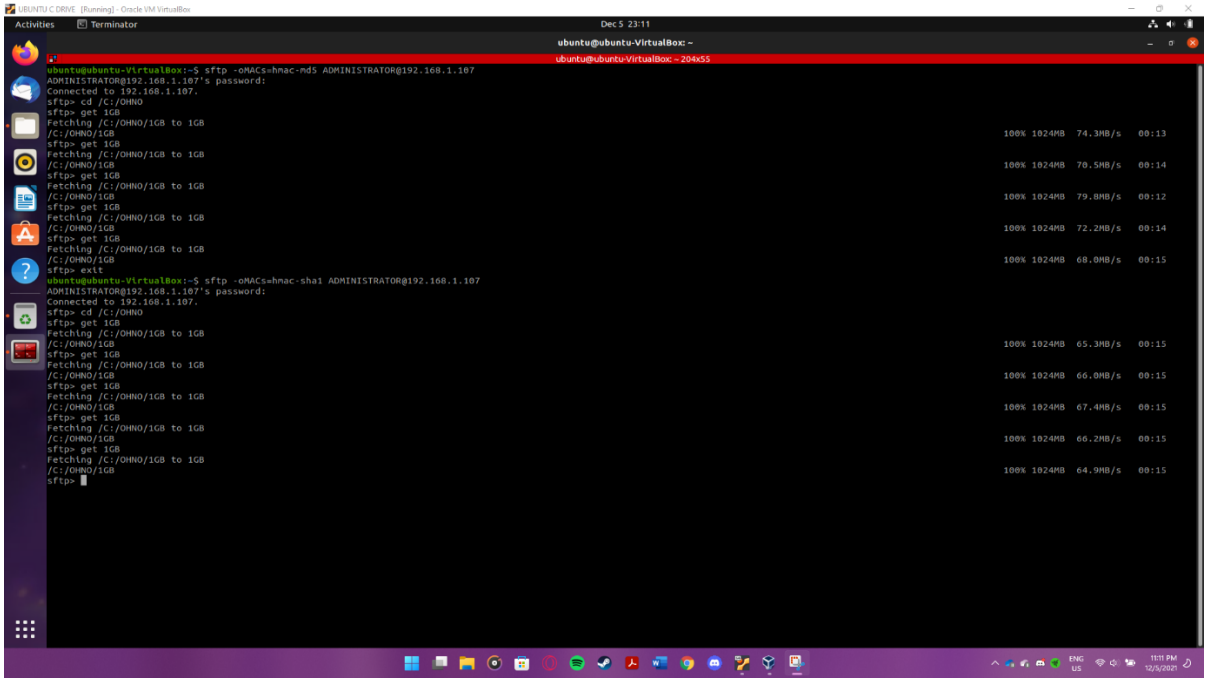

Figure 6 Transfer of 1 GB file

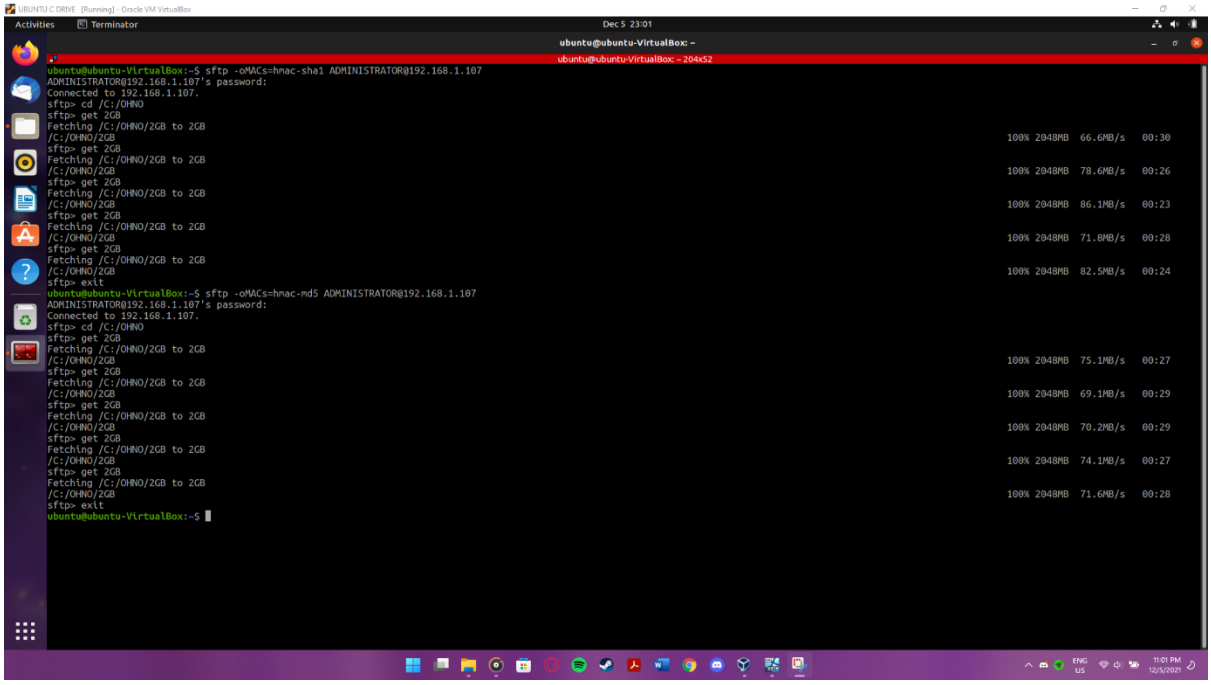

Figure 7 Transfer of 2 GB file

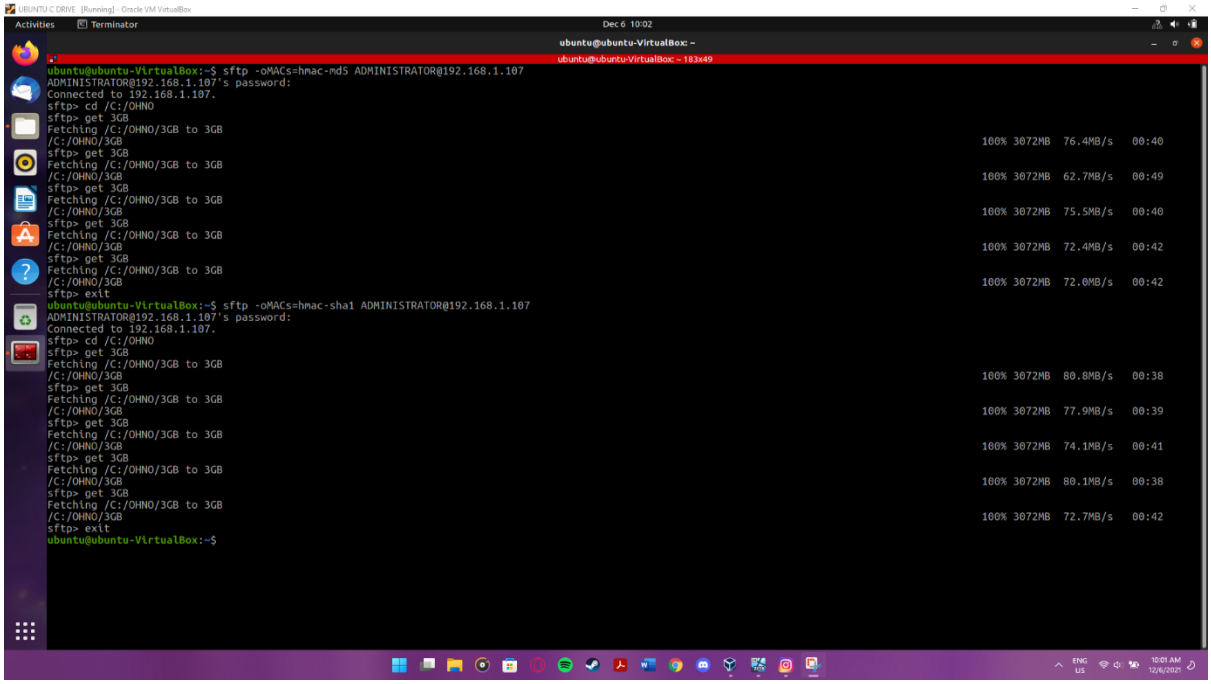

Figure 8 Transfer of 3 GB file

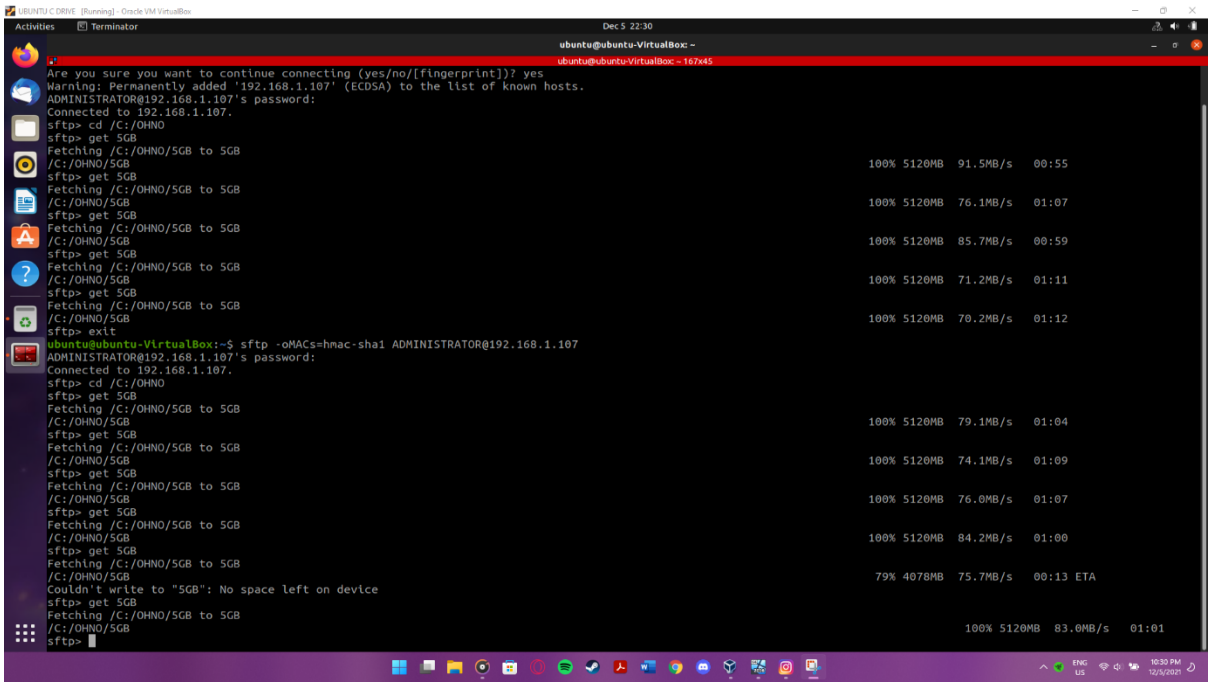

Figure 9 Transfer of 5 GB file

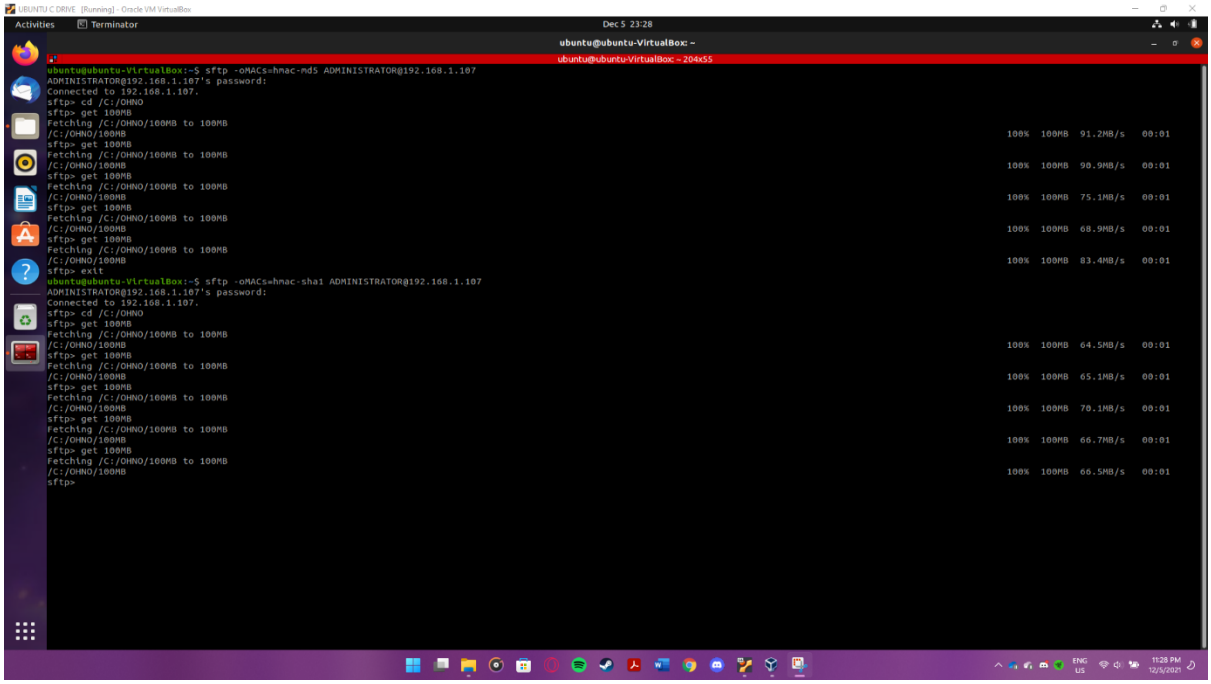

Figure 10 Transfer of 100 MB file

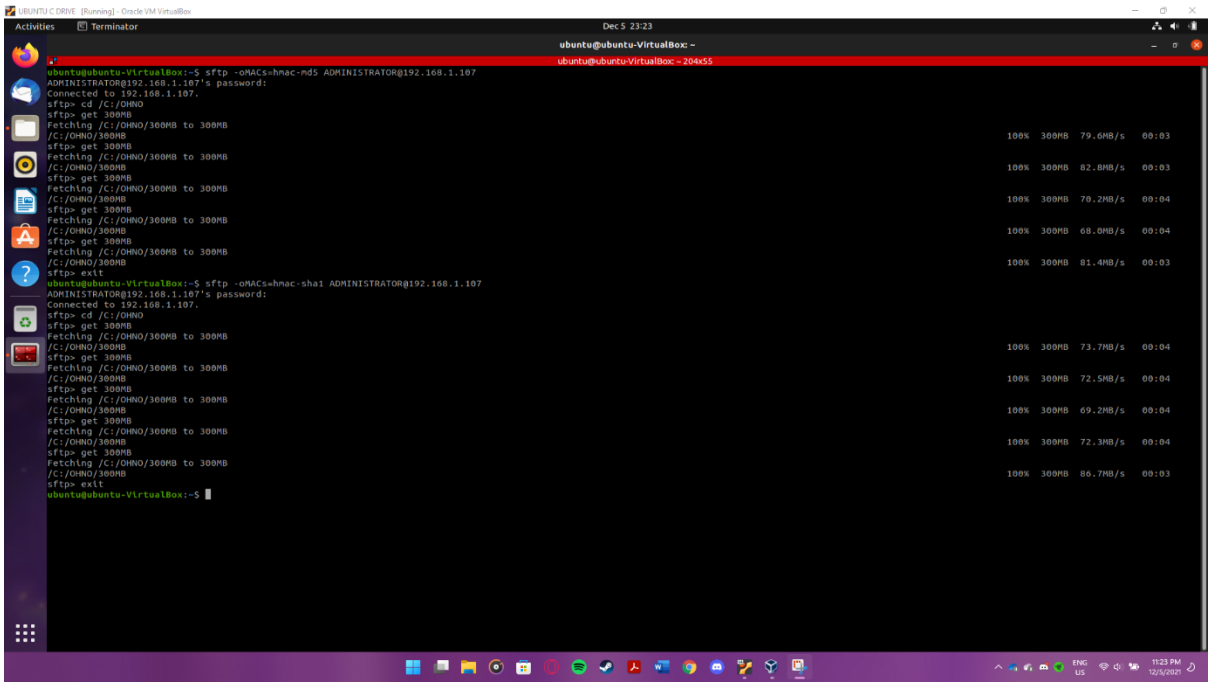

Figure 11 Transfer of 300 MB file

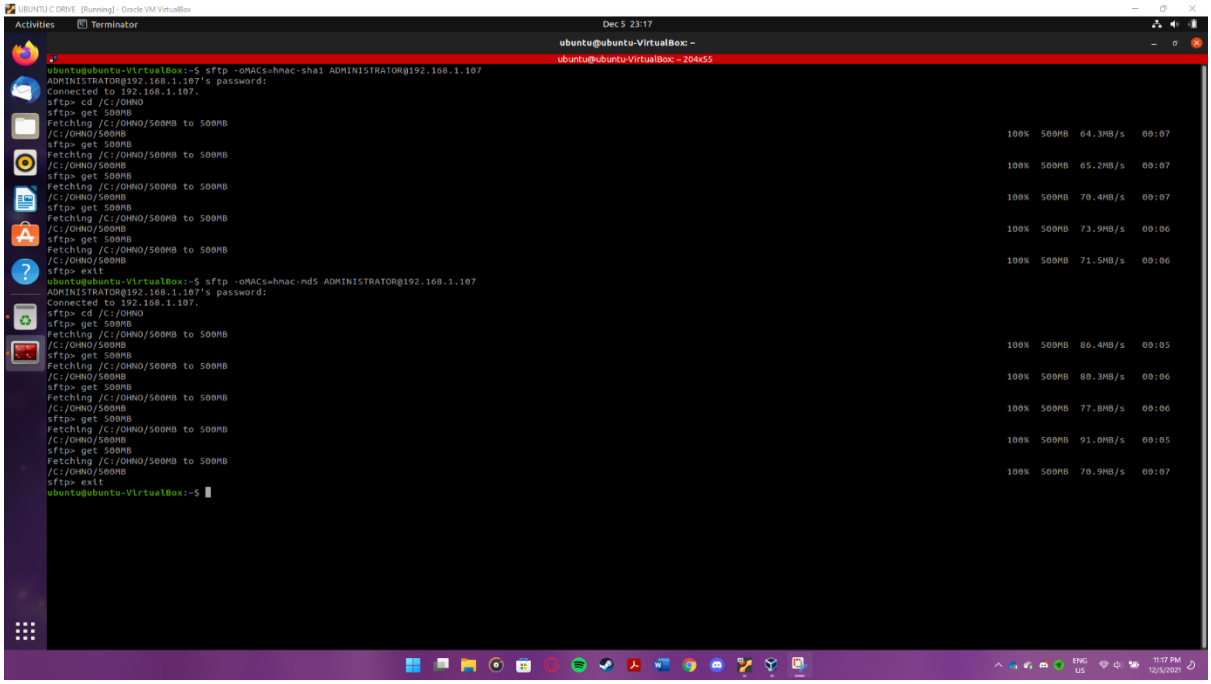

Figure 12 Transfer of 500 MB file