Lab #3 by William Lentini, Shamar Roberts, and Daniel Lake

1. Inspect the newly created file. Does the digest here change significantly from the digest contained in the doi-SHA-256.hash file? Yes they are different the texts in both are completely different

2. Experiment feeding the doi.txt file as the Message file and the doi-altered-SHA-256.hash file. What do you get when you try to verify? I get a message saying hash verification failed

Applying Hash functions towards active adversaries. Now that you've seen how Hash Functions work, think of a scenario, and elaborate in your Lab Report, how this weak sense of Integrity can be broken when we are operating in an active attacker setting. Recall that an attacker is considered active, if he intercepts messages and has the ability to replace messages which are in transit from a sender receiver. Please give me a clear example to show your understanding.

This weak sense of integrity can be easily broken if the attacker is active. An active attacker attempts to make changes to data within a Network. Hashes by themselves are not strong and do not uphold integrity, they are more of a helper function and need to be paired with something else to be useful in upholding integrity. Hashes produce a fixed length output, Integrity is defined as the ability of two communicating parties to detect alterations of messages between each other. Hash algorithms are susceptible to active attackers because an attacker can modify a document so that it has the desired hash value.

1. Inspect the created file, see what's in it. How large is it etc.? Hint: If you click at the end of the string, Notepad will tell you the next usable column, from which

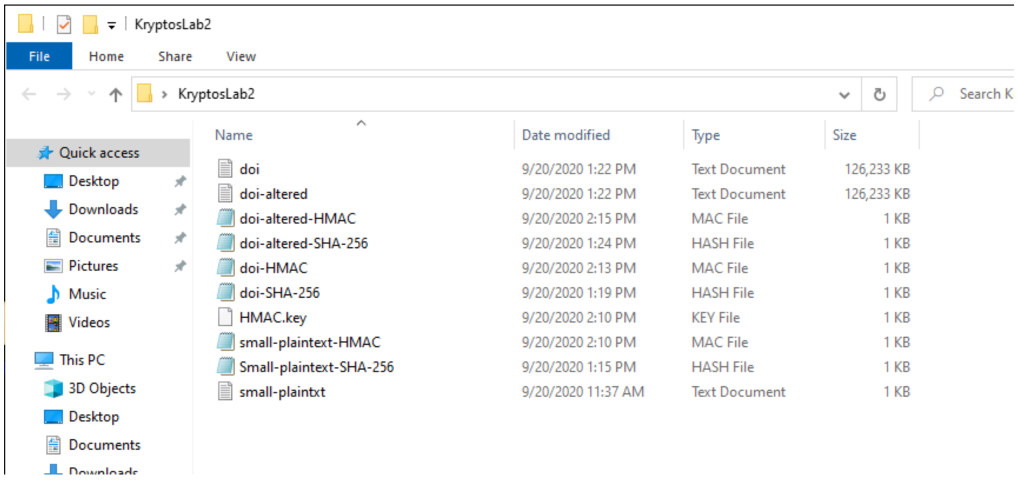
you can deduct 1. Use the Notepad editor to inspect it and take screenshots of this stage. 16 bits

2. Try to feed the doi-altered.txt file under the same key. What will the created tag file (call it doi-altered-HMAC.mac) will look like? Will it differ from the doi-HMAC.mac file? Yes they will not look alike

3. Experiment with doi.txt as the message file and the doi-altered-HMAC.mac file as the tag file. What output do you get when you try to verify? Verification Fail

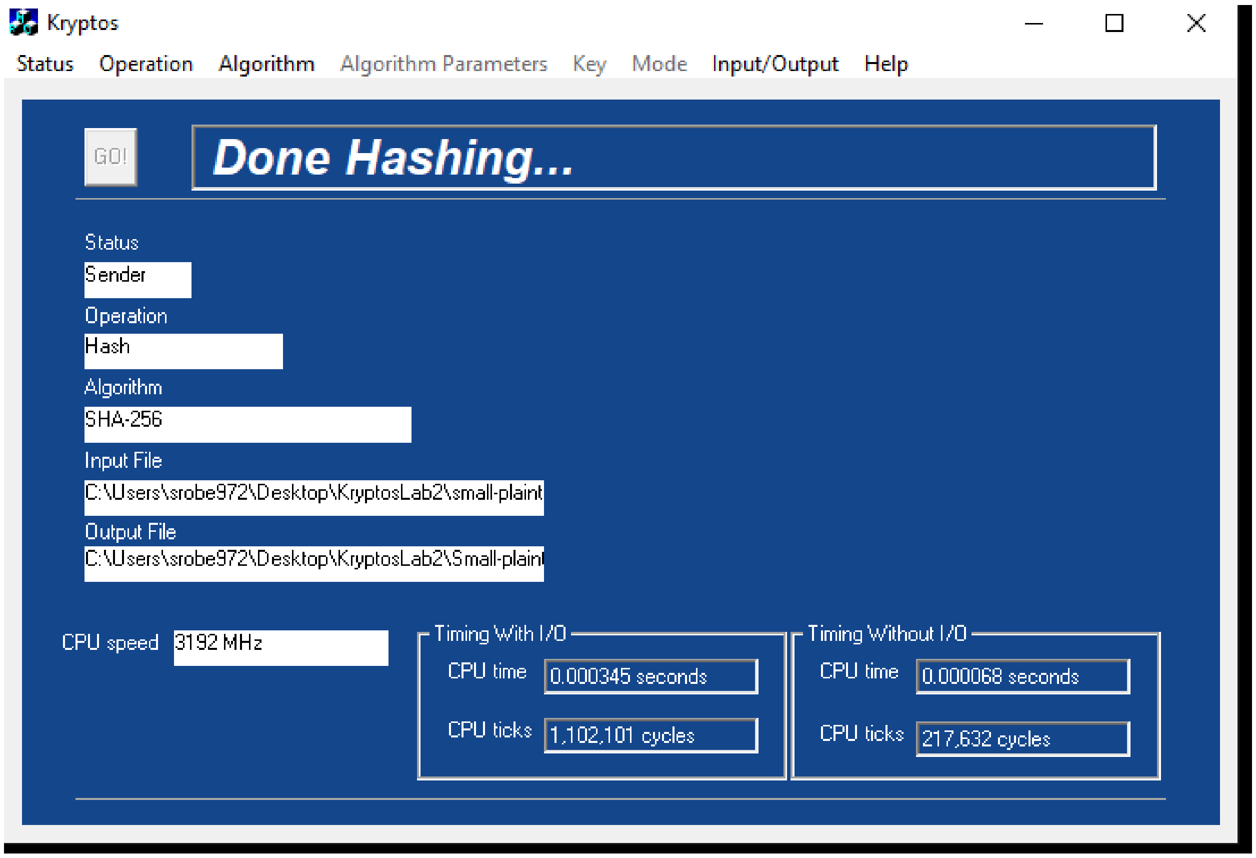
Give me a good coherent point of view, why HMACs would be successful in preventing active attacks with respect to Integrity and why a Hash function won't? Please give me a clear example to show your understanding.

An HMAC would be more successful in preventing an active attack compared to hashes. This is because hashes do not require a key meaning anyone can end up editing the document, on the other hand MACs are required to use a key. And to keep up integrity you would share your key to a trusted party. This means that an active attacker would not be able to intercept the message and change it because they do not have the specific key that is required to edit the message. This makes HMAC's a great tool when it comes to authenticity and integrity because you know who you are sharing your message with and know that there is a low chance of your message getting intercepted and changed.



Name	Date modified	Type	Size
doi	9/20/2020 1:22 PM	Text Document	126,233 KB
doi-altered	9/20/2020 1:22 PM	Text Document	126,233 KB
doi-altered-HMAC	9/20/2020 2:15 PM	MAC File	1 KB
doi-altered-SHA-256	9/20/2020 1:24 PM	HASH File	1 KB
doi-HMAC	9/20/2020 2:13 PM	MAC File	1 KB
doi-SHA-256	9/20/2020 1:19 PM	HASH File	1 KB
HMAC.key	9/20/2020 2:10 PM	KEY File	1 KB
small-plaintext-HMAC	9/20/2020 2:10 PM	MAC File	1 KB
Small-plaintext-SHA-256	9/20/2020 1:15 PM	HASH File	1 KB
small-plaintxt	9/20/2020 11:37 AM	Text Document	1 KB

- Screenshot of all files created during Lab 3



Done Hashing...

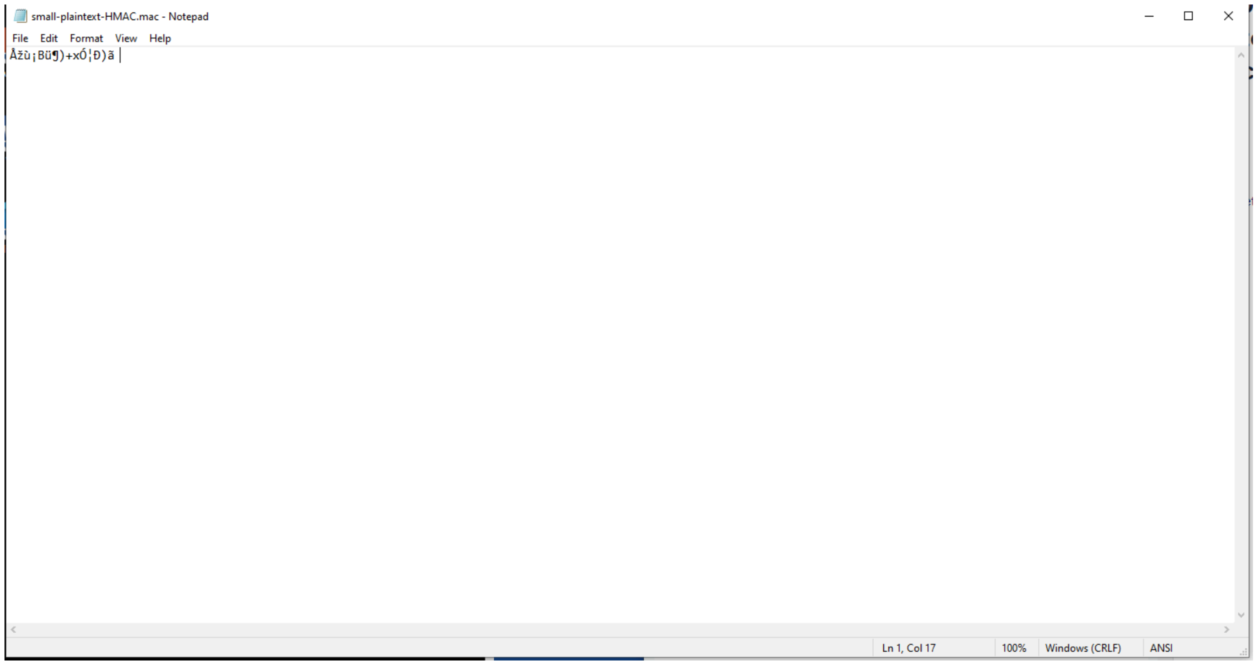
Status: Sender
Operation: Hash
Algorithm: SHA-256
Input File: C:\Users\srobe972\Desktop\KryptosLab2\small-plaint
Output File: C:\Users\srobe972\Desktop\KryptosLab2\Small-plaint

CPU speed: 3192 MHz

Timing With I/O
CPU time: 0.000345 seconds
CPU ticks: 1,102,101 cycles

Timing Without I/O
CPU time: 0.000068 seconds
CPU ticks: 217,632 cycles

- Screenshot of Hashing for the small-plaintext file

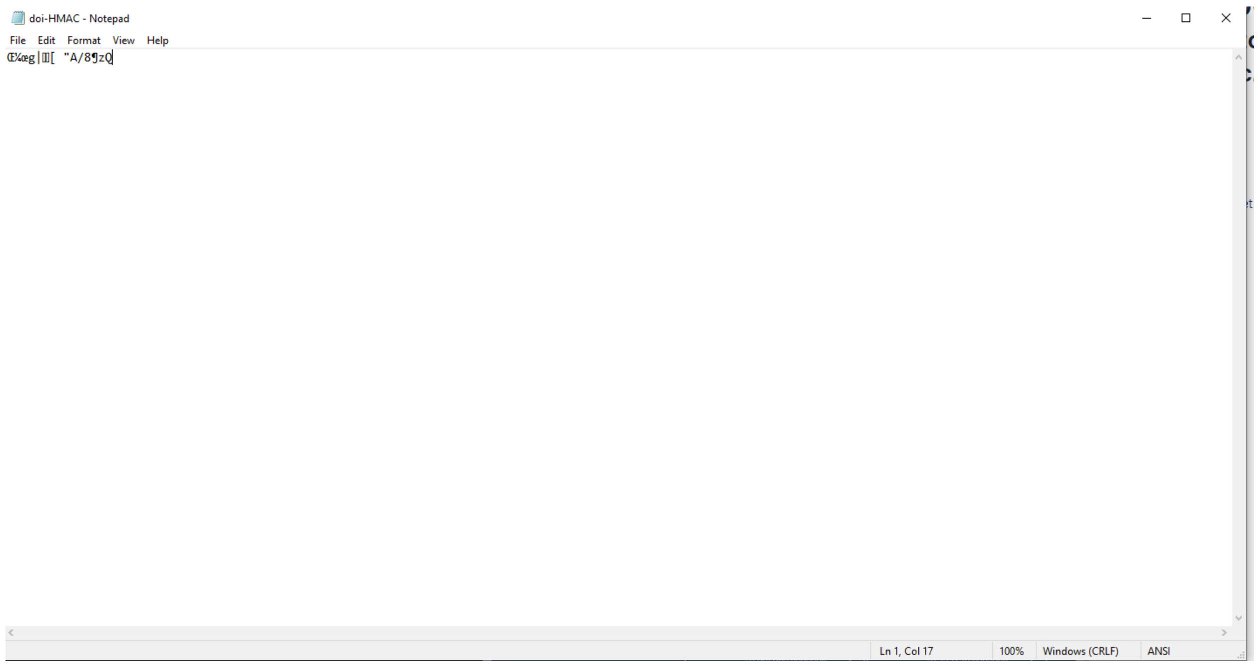


small-plaintext-HMAC.mac - Notepad

```
Åžù¡Bü¶)+xÓ¦Ð)ã 
```

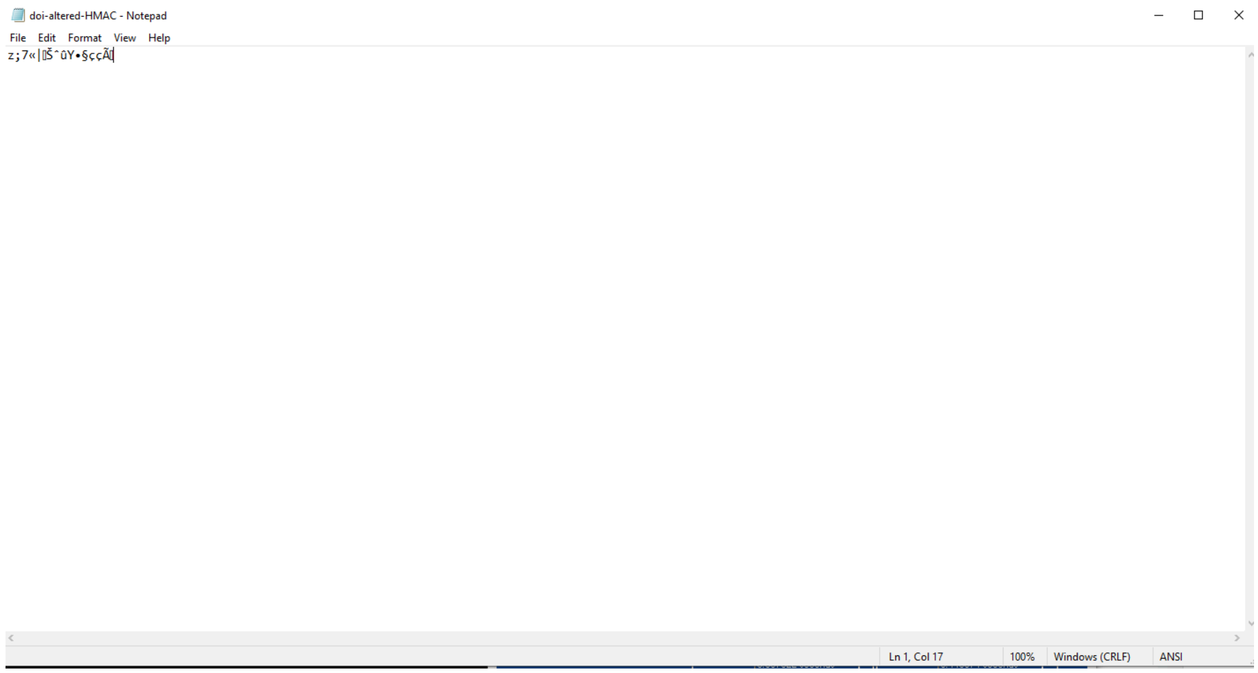
Ln 1, Col 17 | 100% | Windows (CRLF) | ANSI

- Screenshot of Small-plaintext HMAC file the length of this file is 16



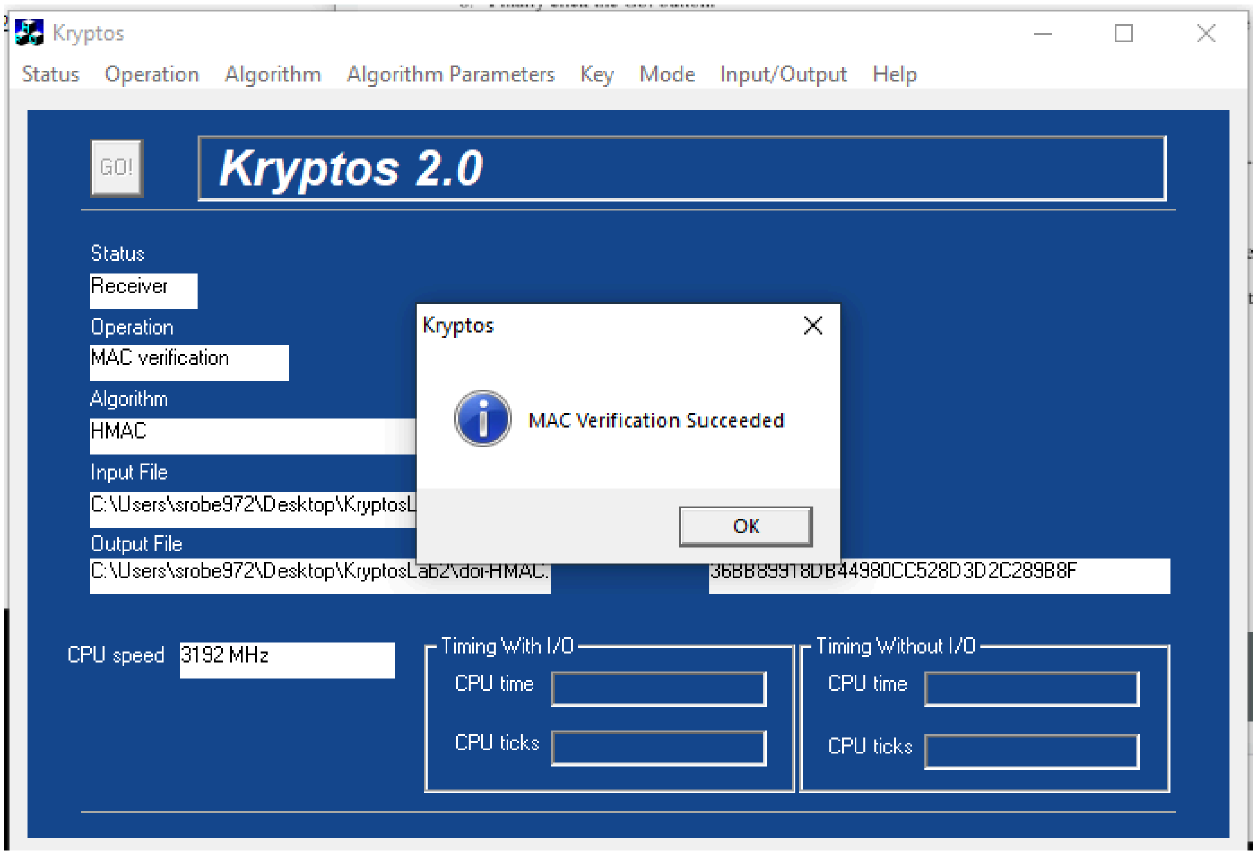
```
Œ¼œg|▯▯[  "A/8¶zQ
```

- HMAC Generated for DOI file



```
z;7«|▯Šˆ ûY•§ççÃ▯
```

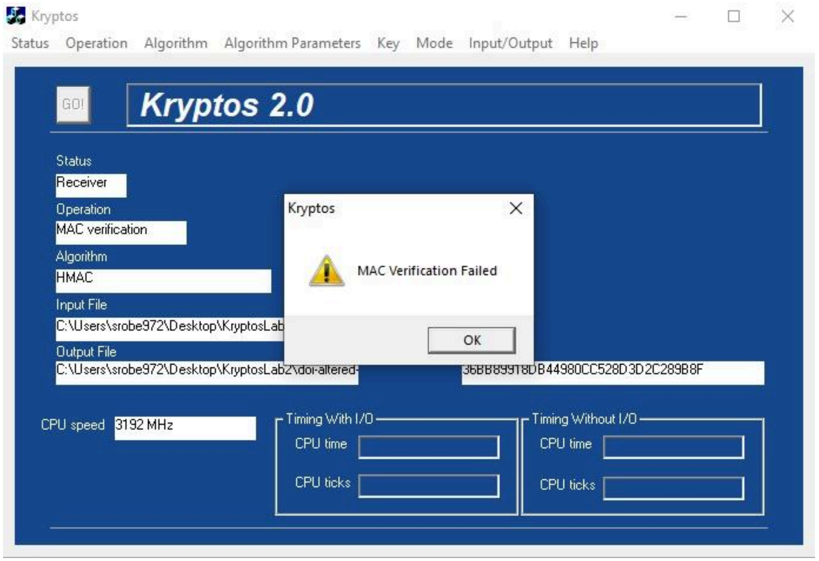
- HMAC generated for the DOI altered file even though one letter was changed the message looks completely different


Kryptos
Status Operation Algorithm Algorithm Parameters Key Mode Input/Output Help
GO!
Kryptos 2.0

Status
Receiver
Operation
MAC verification
Algorithm
HMAC
Input File
C:\Users\srobe972\Desktop\KryptosL
Output File
C:\Users\srobe972\Desktop\KryptosLab2\doi-HMAC.
36BB89918DB44980CC528D3D2C289B8F
CPU speed 3192 MHz
Timing With I/O
CPU time
CPU ticks
Timing Without I/O
CPU time
CPU ticks

Kryptos
MAC Verification Succeeded
OK

HMAC verification for the DOI file which ended up being successful


Kryptos
Status Operation Algorithm Algorithm Parameters Key Mode Input/Output Help
GO!
Kryptos 2.0

Status
Receiver
Operation
MAC verification
Algorithm
HMAC
Input File
C:\Users\srobe972\Desktop\KryptosLab
Output File
C:\Users\srobe972\Desktop\KryptosLab2\doi-altered-
36BB89918DB44980CC528D3D2C289B8F
CPU speed 3192 MHz
Timing With I/O
CPU time
CPU ticks
Timing Without I/O
CPU time
CPU ticks

Kryptos
MAC Verification Failed
OK

- MAC verification failed when tried entering DOI-HMAC for input and DOI-Altered for Output