# Cryptography Homework 8: Hashes

Both Linux and Windows have applications that can compute hashes. This homework will give you basic familiarity with them.

Software download sites often supply hashes of the programs they provide so you can verify the download is correct. You should consider a correct hash as proof the program was downloaded without corruption. It does not prove that the software has not been tampered with, however. If an attacker has compromised the download site to upload modified software, they can also upload the hash of the modified software. To verify the software has not been improperly modified, you need either a digital signature (signed by a key or PKI you trust) or a hash that’s stored in a separate, secure location.

## Hashes in Linux

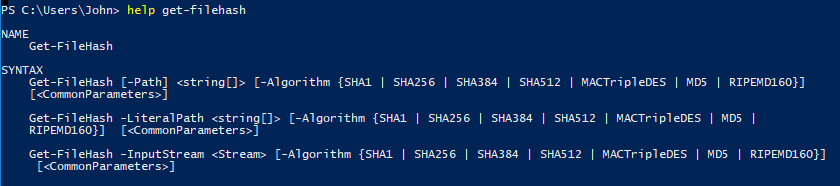
Linux has several hashing applications that are installed by default. Generally, the name is the hash type with “sum” appended to it. For example: md5sum (128 bits long), sha1sum (160 bits), sha256sum (256 bits), and sha512sum (512 bits). Although md5 and sha1 are no longer secure against collision attacks, they are still useful for verifying that a file was not corrupted during download. You can get information on how to use the hash applications with the --help option or by using man pages (man sha512sum, for example.)

SHA-3 hashing applications are available, but not usually installed by default. They can be installed using “yum install sha3sum” on RedHat variants. In Ubuntu “sudo apt install libdigest-sha3-perl”.

You can take hashes of files, “sha256sum ~/filename”, or pipe input into it,  
[other commands] | sha256sum

## Hashes in Windows

You can download versions of md5sum, sha1sum, etc. for Windows, but the simplest way is to use the Get-FileHash commandlet in PowerShell.



The Get-FileHash commandlet does not accept input from the pipeline and does not compute SHA-3 hashes.

## Homework

### Compute the SHA-512 hash of a file (of your choice) in both Windows and Linux

Paste the results here.

### SHA-1 hash collision

Google used the files good.pdf and bad.pdf to demonstrate a SHA-1 collision (<https://shattered.io/static/shattered.pdf>), and they are in Canvas. Compute the SHA-1 hashes of the two files. Are they the same? Do the files look the same? Also compute the md5 and sha256 hashes--are they the same? The files shattered1.pdf and shattered2.pdf are another pair of files that demonstrate a successful hash collision against SHA-1.

Paste your answers here.

### Linux gotcha with echo and hashes

It is tempting to take the hash of a short string by echoing it into the hash application, as in:  
echo "this is some string " | sha256sum  
However, by default echo adds a newline character to the end of all strings. This can cause confusion, especially with painful wrong answers on Capture the Flag questions. The -n option in echo will suppress the additional trailing newline character.

Compute the SHA-256 hash of a string of your choosing with  
echo "your choice " | sha256sum

Paste your hash here.

Compute the SHA-256 hash of a string of your choosing with echo and the -n option  
echo -n "your choice "| sha256sum

Paste your hash here.

The xxd application is useful for showing the hex of a file or input string. You can use it to see what echo is doing. Run  
echo "your choice " | xxd  
and then  
echo -n "your choice " | xxd.  
What’s different between the two?

Note: Another gotcha can happen when you cut and paste from Word into Linux. Word likes to change quotes to ‘smart quotes’, which causes problems. I try to change all the smart quotes back to quotes in places where you might cut and paste, but you never know…