# INFO3068 Security Activity 8: Digital Signatures

Complete the blanks below and submit this file to the submission folder in FanshaweOnline.

1. This lab assumes you already have the OpenSSL command line utility installed. Alter the folder names as needed.
2. Create a simple text file, where XX are your initials:

**notepad myfileXX.txt**

Edit and save.

1. Run this command:

**openssl dgst –md5 myfileXX.txt**

1. What is the output?

MD5(myfileHI.txt)= 840b7dff67a21e78f707001395a2e718

1. What type of algorithm is MD5?

It is a cryptographic hash algorithm used to generate a 128-bit digest from a sting of any length

1. Should we use MD5? Why not?

some pros of using the MD5 algorithms are that it is easier to compare and store small hashes and it is useful when you have to compare files or code to identify any types of changes

1. Let’s try something more realistic:

**openssl dgst –sha256 myfileXX.txt**

1. What do you notice about the output?

SHA256=(myfileHI.txt)= 66a812868f104c9879374e8afaa939a645289db33be24cded01e81193e3fe217

1. Let’s edit the myfileXX.txt file and run the command again. Did the output change? Is that what we expect?

SHA256(myfileHI.txt)= c2ce54bb504a249e1a7349a65bc70a033925ff48048d0005d0cb36f1574ce4e6

When I change the text in the file the output did change, when I did with sha256. If we changed the text then since there is new data in the file, so the command should give us new values

1. For this next part, we will need an asymmetric encryption key. We can create a public-private key pair, along with a self-signed digital certificate:

**openssl req -x509 –nodes –days 365 –sha256 -newkey rsa:2048 –keyout mykey.pem –out mycert.pem**

1. Let’s encrypt the digest with an encryption key:

**openssl dgst –sha256 -sign mykey.pem -out myfileXX.txt.sha256 myfileXX.txt**

1. If we encrypt a message digest with a private key, what do we end up with? (Hint: see title of this lab!)

A Digital Signature

1. We’ll next to extract the public key from the key file:

**openssl rsa –in mykey.pem –pubout > mykey.pub**

1. Let’s verify the signature with a public key:

**openssl dgst –sha256 -verify mykey.pub -signature myfileXX.txt.sha256 myfileXX.txt**

1. Is the verification OK?

Yes, “Verified Ok”

1. Modify the myfileXX.txt file again.
2. Re-run the verification. Did it work? Why or why not?

I got a verification error “Verification failure”, I’m pretty sure it failed because the verification file, is for the first txt file, since the data in the file changed it would not be a proper verification

1. Is that what we expected?

Yes, since the file we are verifying is changed, it should not be correctly.