# What is OpenSSL?

OpenSSL is an open source cryptographic toolkit that implements. Tools are available to carry out key generation and storage, encryption and authentication with a selection of popular algorithms, creation and management of digital certificates, and various other related activities. OpenSSL is available from http://www.openssl.org/

OpenSSL tools are available at the command line and as an application programming interface (API).

# How will we use OpenSSL?

We will use the command-line OpenSSL tools in this course for various practical exercises in encryption and authentication. OpenSSL comes pre-installed with most versions of Linux and Mac OS X.

A positive side effect of these practical exercises using OpenSSL is that you will get to gain experience with shell commands on Linux. In room D04 Ubuntu is the Linux OS installed and so we can use OpenSSL on those machines easily.

If you wish to use OpenSSL at home or in a BYOD lab you will need to have access to either a version of Linux or a MAC OS X.

# OpenSSL Basics

You can execute OpenSSL commands directly from the Linux prompt using the general command format:

$ openssl command [options]

Depending on the command, there may be zero or more options

For example, you could enter:

$ openssl version

OpenSSL 1.0.1e 11 Feb 2013

You can get a list of OpenSSL commands by entering any invalid command:

$ OpenSSL > help

openssl:Error: 'help' is an invalid command.

Standard commands

asn1parse ca ciphers cms

crl crl2pkcs7 dgst dh

dhparam dsa dsaparam ec

ecparam enc engine errstr

<etc>

Message Digest commands (see the `dgst' command for more details)

md4 md5 mdc2 rmd160

sha sha1

Cipher commands (see the `enc' command for more details)

aes-128-cbc aes-128-ecb aes-192-cbc aes-192-ecb

aes-256-cbc aes-256-ecb base64 bf

bf-cbc bf-cfb bf-ecb bf-ofb

camellia-128-cbc camellia-128-ecb camellia-192-cbc camellia-192-ecb

camellia-256-cbc camellia-256-ecb cast cast-cbc

cast5-cbc cast5-cfb cast5-ecb cast5-ofb

des des-cbc des-cfb des-ecb

<etc>

# Symmetric encryption exercise using OpenSSL

To complete an encryption exercise using OpenSSL there are a few items we need to assemble first:

1. A plaintext file to encrypt
2. An encryption key
3. An encryption algorithm

**Step 1** Create a file containing the following text and name it myplaintext.txt

***A dreaded sunny day, So I meet you at the cemetery gates***

***Keats and Yeats are on your side, While Wilde is on mine***

Verify the contents of your text file

$ ls

$ cat myplaintext.txt

**Step 2** In the same directory, use OpenSSL to create a file containing a pseudo random number 8 bytes long. Later, you will use that number as your DES key. Enter:

$ openssl rand -out des64\_key 8

$ ls

$ cat des64\_key

(some unprintable characters will be displayed as the key is binary)

**Step 3** Now use the DES encryption algorithm to encrypt your file

$ openssl des -e -kfile des64\_key -in myplaintext.txt -out myciphertext.bin

Next check your current directory listing. You should see the newly encrypted file there named myciphertext.bin.

$ ls

Have a look at the contents of the encrypted file. What do you see?

$ cat myciphertext.bin

# Symmetric decryption exercise using OpenSSL

To complete a decryption exercise using OpenSSL there are a few items we need to assemble first:

1. A ciphertext file to decrypt

2. A decryption key (the same key used to encrypt)

3. A decryption algorithm (the same algorithm used to encrypt)

To decrypt the encrypted file that you just created, you must be in the same directory where you have the key saved and the ciphertext file.

$ ls

$ openssl des -d -kfile des64\_key -in myciphertext.bin -out decryptedfile.txt

Next check your current directory listing again. You should see the newly decrypted file there named decryptedfile.txt

$ ls

Check the contents of the file, are they the same as the original?

$ cat decryptedfile.txt