# Symmetric Encryption

Encryption Algorithms can be either:

* Symmetric Encryption Algorithms
  + Secret key encryption
  + same key used on both sides to encrypt and decrypt
* Asymmetric Encryption Algorithms
  + Public key encryption
  + Different key pairs used to encrypt and to decrypt

We will use OpenSSL to complete Symmetric encryption exercises in the this practical.

# Symmetric Encryption exercises

## Preparation

1. You should sit beside someone that you can work with as you will need a partner to complete some of the exercises.
2. You should probably create a folder to store your files in and navigate to that folder.
3. To complete the encryption exercises you will need to first prepare the following:
   1. A selection of Encryption keys.
      * Use a random number generator to create 3 keys of differing length. Save them in the directory created in the last step. Here is an example of creating a 16-byte key and naming it key16.

$ openssl rand -out key1 16

* 1. A selection of Plaintext files which you can use to practise encryption on.
     + Use the Linux editor mentioned last week to create 2 plaintext files containing some text. Save them in the directory where you saved the keys created in the last step.
  2. Three different encryption algorithms supported by OpenSSL.
     + Use OpenSSL documentation/help to choose 3 algorithms.

## Encryption

1. Encrypt both of your plaintext files using each of the three algorithms chosen.

Here is an example of a file named plaintext.txt being encrypted with the AES algorithm using key16. The ciphertext output is stored in a file named ciphertext.bin.

$ openssl aes-128-cbc -e -kfile key16 -in plaintext.txt -out ciphertext.bin

## Decryption

1. To practice decrypting files each student should exchange **keys** and **ciphertext files** with each other. This can be done over email or USB drive. Make sure you find out which algorithms were used by your partner.
2. Once you have received encrypted files from your partner, try to read/edit the files and see if they make any sense?

$ cat ciphertext.bin

1. Now you should decrypt the encrypted files. Here is an example of a file named ciphertext.bin being decrypted with the AES algorithm using key16. The ciphertext output is stored in a file named decrypted.txt.

$ openssl aes-128-cbc -d -kfile key16 -in ciphertext.bin -out decrypted.txt

1. Try to read the contents of the decrypted files.

$ cat decrypted.txt

1. On Moodle, you will find an encrypted file and the key required to decrypt it. Decrypt the file and find the message inside.

# Effect of Compression on Encryption

We want to see the effect of compression on Encrypted files. Let’s create a large text file (10 MB) and then experiment by:

* Viewing the size of the original large file
* Compressing the large file and then viewing its size
* Encrypting the large file, compressing the resulting encrypted file and then viewing its size

1. Create a new file, this time with a lot of English language words or similar (say about 10MB in total). You can name it whatever you want.
2. Encrypt it using an algorithm and key of your choice.
3. Check the size of the encrypted file and compare with the original.
4. Compress both the original text file and the encrypted file. Use any popular general-purpose compression program such as zip; e.g.

$ zip bigplaintext.zip bigplaintext.txt

$ zip bigciphertext.zip bigciphertext.bin

What do you notice when comparing the file sizes? Which resulting zip file is bigger? Why is this the case?

# Performance Testing

Encryption and decryption take time, especially on slow processors. In this section, you will benchmark the performance of various encryption algorithms on your machine.

1. Create or locate a large test file of several MB, say 100MB.
2. Record how much time it takes to encrypt it using various encryption algorithms. Choose at least 5 algorithms, including DES, triple DES (3DES), RC4 and AES with at least two different key lengths. To do this, use the time command, which is simply prefixed to the command you are running, like this:

$ time openssl des -kfile key2 -in plaintext.zip -out des\_encrypted.bin

real 0m0.736s

user 0m0.576s

sys 0m0.057s

(user time is what matters)

Ideally, take each time measurement a few times (say 3 times) and exclude any that are noticeably slower than the others (to reduce impact of short-term processor activity from running programs).