# INFA723 Cryptography and Network Security

# Lab6 Use OpenSSL to Create RSA Public/Private Key (512bits) without Password Protection

# 1 Introduction

OpenSSL toolkit provides a command line tool and a crypto library used for a wide range of crypto operations. In this exercise, we are going to learn how to use the command line tool to create RSA public key and private key without password protection. The lab will further demonstrate how to use OpenSSL to encrypt/decrypt a file using an RSA public/private key. Note that for all the functions implemented in the lab, you can find a way to implement them using the OpenSSL crypto library.

Note that the lab has a limitation of the size of the plaintext since it must be smaller than the modulo n we generated in the lab. If you want to encrypt a large file using the 512bits keys, you have to split the message in small chunks and encrypt each chunk separately.

A lab package has been created for the lab. Go to the class website and download the lab package. Assume your Cygwin installation folder is c:\cygwin. Unzip the package to your Cygwin home folder, e.g., C:\cygwin\home\user name.

# 2 Objectives

* Learn how to use OpenSSL command tool to generate a RSA public key and a private key without password protection
* Learn how to use OpenSSL command tool to encrypt and decrypt a file using RSA public-key cryptography

# 3 Use OpenSSL to Create RSA Public/Private Key (512bits) without Password Protection

1. Generate a 512 bits RSA key.

**$ openssl genrsa -out private512.key 512**

**Generating RSA private key, 512 bit long modulus**

**....................++++++++++++**

**..++++++++++++**

**e is 65537 (0x10001)**

1. Extract RSA public key from the private key.

**$ openssl rsa -in private512.key -pubout -out public512.key**

**writing RSA key**

1. Encrypt a file using RSA public key.

**$ openssl rsautl -encrypt -pubin -inkey public512.key -in plaintext1.txt -out encrypted.txt**

Note that the OpenSSL RSA utility does not automatically separate the message to make sure m<n. Thus, the size of the plaintext here cannot be too long. If the plaintext is too long, an error will happen when encrypting the file.

1. Decrypt a file using RSA private key.

**$ openssl rsautl -decrypt -inkey private512.key -in encrypted.txt -out c.txt**

1. Compare the difference between plaintext1.txt and c.txt

**$ diff plaintext1.txt c.txt**

# 4 Question

1. (8 points) In Step 1, what is the public key (e, n) used? Why?
2. (12 points) A certificate, Amazon.cer, can be found in the lab 6 folder, answer the following questions:
   1. What is the signature hash algorithm used to create the certificate?
   2. What is the public key cryptography used to create the signature?
   3. What is the size of the public key?
   4. What is the *e* used in the public key in the certificate?
   5. Who issued the certificate?
   6. Why can you trust the certificate?