**Description**

The objective of this lab is to get you familiar with common cryptographic algorithms.

**Part 1: Install OpenSSL and understand how it works**

1. Install Visual C++ 2008 if not already installed. (<http://www.microsoft.com/downloads/details.aspx?familyid=9B2DA534-3E03-4391-8A4D-074B9F2BC1BF>).
2. Install OpenSSL (<http://slproweb.com/download/Win64OpenSSL_Light-1_0_1e.exe>). You might need to restart the VM.
3. Launch OpenSSL
   * Launch cmd
   * Go to “C:\OpenSSL-Win64\bin”
   * Launch openssl
4. What does the ***enc*** command do?

**Part 2: Use OpenSSL for Symmetric Encryption**

1. Encrypt “eme.txt” using DES to produce “dme.txt”; the key is “password”.
2. Decrypt “dme.txt” to “emed.txt”, compare “eme.txt” to “emed.txt”.
3. Encrypt “eme.txt” using DES to produce “dme.txt”; the key is password, and use a salt value of your choice.
4. Encrypt “eme.txt” using DES in CBC mode to produce “dme-cbc.txt”; same key.
5. Encrypt “eme.txt” using DES in EBC mode to produce “dme-ebc.txt”; same key.
6. Encrypt “eme2.txt” using DES in EBC mode to produce “dme2-ebc.txt”; same key.
7. Compare “dme-ebc.txt” and “dme2-ebc2.txt”.
8. Encrypt “eme.txt” using RC4 with a 40bit key, decrypt and check the output
9. Create a file called “pass.txt” that contains “password” and use it to carry out sub-question c.

**Part 3: Use OpenSSL for Asymmetric Encryption**

1. What is the PEM format?
2. Generate a RSA pair of keys with e=3 and a modulus of 1024 bits.
3. Use the public key to encrypt “eme.txt”, then decrypt it and compare the output to “eme.txt”.
4. Use the private key to encrypt “eme.txt”, then decrypt it and compare the output to “eme.txt”.

**Part 4: Use OpenSSL for Hashing**

1. Hash “eme.txt” using MD5.
2. Hash “eme.txt” using SHA1.
3. Reproduce the digital signature process as described in the course slides. Sign “eme.txt” using SHA1 and RSA (use the keys you generated in part 3).

**Part 5: Use OpenSSL for Certificates**

1. What is the CSR format?
2. What is the ASN.1 format? How is it used in certificates context?
3. What is the DER format? How is it used in certificates context?
4. Create a self-signed X.509 certificate for the keys generated in part 3.

**Useful links.**

<http://www.openssl.org/docs/apps/enc.html>

<http://farid.hajji.name/blog/2009/07/15/encryption-and-decryption-with-openssl/>