**Georgia Gwinnett College**

**School of Science and Technology**

**ITEC 3300: Information Security**

**OpenSSL Assignment**

**Introduction**

In this assignment you will put together the tools and OpenSSL commands you learned from the lectures and the three OpenSSL labs, and use them to perform data encryption and decryption, as well as signing and verification.

**Instructions**

1. Open the Cisco AnyConnect Secure Mobile Client on your computer and connect to vpn.ggc.edu. Then choose the ITECLAB group, enter your GGC username and password, and press OK to complete the VPN connection.
2. From the application PuTTY or the command prompt, login to 172.20.1.106 by SSH, using *only* your GGC username as **both** the username and password.
3. Copy the file **Assignment.zip** from the directory **/home/yding** to your home directory using the command **cp /home/yding/Assignment.zip .**
4. Unzip **Assignment.zip** using the command **unzip Assignment.zip**, then change directory to **Assignment** using the command **cd Assignment**

**Exercise**

Suppose that you want to communicate securely with a user Bob. You have not shared a secret key with Bob yet. However, Bob has certified and published his RSA public key. Suppose that the file **pubKey.bin** in the directory **Assignment** contains Bob’s RSA public key.

The file named **msg.txt** in the directory **Assignment** contains the message you want to send to Bob. You want to achieve both confidentiality and integrity for the message. The first step is to encrypt the message. As you learned in class, because public-key encryption is slow, it is a common practice to use *hybrid* encryption, namely, use public-encryption to encrypt and exchange a secret key for private-key encryption (like AES), and use private-key encryption to encrypt the data.

1. Generate a random 256-bit key for AES and write it to a file named **aesKey.bin**.
2. Using 256-bit AES with the CBC mode, encrypt **msg.txt** with the key **aesKey.bin** from Step 1, using the option **-pbkdf2** to derive a one-time key and IV, and write the ciphertext to a file named **cipherMsg.bin**.
3. Encrypt **aesKey.bin** using the RSA public key **pubKey.bin** and write the ciphertext to a file named **cipherSK.bin**. Take a screenshot of the **three commands from Steps 1 – 3**. Include the image here.

Now you have achieved confidentiality for the data and key after encrypting **msg.txt** and **aesKey.bin**. Next you want to add integrity to it using RSA digital signature. The first step is to generate your keys for signing and verification.

1. Generate a pair of RSA keys, using **3072 bits** as the key length, and write the keys to a file named **sigKey.bin**.
2. Extract the public verification key from **sigKey.bin** and write it to a file named **verKey.bin**.
3. Sign *both* **cipherMsg.bin** and **cipherSK.bin** from Steps 2 and 3, using the hash function **sha512**, and write the signatures to files named **sigMsg.bin** and **sigSK.bin**, respectively. (Which key should you use for signing)?

Take a screenshot of the **four commands from Steps 4 – 6**. Include the image here.

Now you have achieved both confidentiality and integrity for the data and key, using the encrypt-then-sign approach. Suppose that you have also certified and published your public verification key **verKey.bin** for everyone to verify your signatures. It is time to send the ciphertexts and signatures to Bob.

1. Copy the files **cipherMsg.bin**, **cipherSK.bin**, **sigMsg.bin**, **sigSK.bin** and **verKey.bin** to the directory **Bob** using the command **cp**. Change directory to **Bob** and list the directory.

Take a screenshot of your commands and include the image here.

In the directory **Bob**, the file **rsaKey.bin** is the pair of RSA keys that contains the matching private decryption key to **pubKey.bin**. Suppose that Bob has received your ciphertexts and signatures, and also looked up your public verification key **verKey.bin**.

1. Using proper **OpenSSL** commands, verify the integrity of the encrypted data and key Bob receives from you, and decrypt the ciphertexts to uncover the content of the file **msg.txt**. There should be **four commands** in this step. Take a screenshot of the **four commands** and include the image here.

**Deliverable**

Upload this file, including all the images, to D2L after you complete the assignment.