Implementation of RSA

I chose to use java for this implementation largely because I had no joy using the Swift public & private key generators with the deprecated SecKeyGeneratePair( ), SecKeyEncrypt( ), SecKeyDecrypt( ) fucntions… I will fix this some day!

Anyway so Java implementation is as follows:

1. Key generation (public & private using a KeyPairGenerator instance & its built in functions).

```
//Instantiate the RSA key generator with specified byte size
KeyPairGenerator keyPairGenerator = KeyPairGenerator.getInstance("RSA");
keyPairGenerator.initialize(2048);

//Use keyPairGenerator to generate the KeyPair (public & private)
KeyPair keyPair = keyPairGenerator.generateKeyPair();

// Get the public and private key
PublicKey publicKey = keyPair.getPublic();
PrivateKey privateKey = keyPair.getPrivate();
```

2. Isolate the generated keys into respective variables
3. Call my encryption function passing in the original message and public key – because in RSA we encrypt outgoing messages with the public key
4. Then show the user the encrypted data
5. Call the decryption function passing in the encrypted byte array with the private key – because we decrypt using the private key in RSA completing my use of the **asymmetric cryptographic algorithm.**

```
//Perform encryption process using public key
byte[] cipherByteArray = encryptFunc(originalMsg, publicKey);

//Perform decryption using private key
String decryptedText = decryptFunc(cipherByteArray, privateKey);
```

To elaborate on the encrypt/ decrypt functions

A Cipher instance is used to format our RSA encryption setup, and this instance is initialised with the required encryption /decryption modes & public/ private keys respectively as in screenshot below.

```
public static byte[] encryptFunc (String originalMsg, PublicKey publicKey ) throws Exception
{
    //Get required Cipher Instance & initialise stating mode and public Key
    Cipher cipherInstance = Cipher.getInstance("RSA/ECB/PKCS1Padding");
    cipherInstance.init(Cipher.ENCRYPT_MODE, publicKey);

    //Use cipherInstance doFinal function to encrypt & return byteArray
    byte[] cipherByteArray = cipherInstance.doFinal(originalMsg.getBytes()) ;
    return cipherByteArray;
}

public static String decryptFunc (byte[] cipherByteArray, PrivateKey privateKey) throws Exception
{
    ////Get required Cipher Instance & initialise stating mode and private Key
    Cipher cipherInstance = Cipher.getInstance("RSA/ECB/PKCS1Padding");
    cipherInstance.init(Cipher.DECRYPT_MODE, privateKey);

    //Use cipherInstance doFinal function to encrypt
    byte[] decipherByteArray = cipherInstance.doFinal(cipherByteArray);

    //Convert byteArray to string
    return new String(decipherByteArray);
}
```

To Run this RSA program simply do as follows:

```
RSA_Implementation — -bash — 74×21
MBP2697:RSA_Implementation stephen2697$ javac *.java
MBP2697:RSA_Implementation stephen2697$ java RSA
Enter your message to be encrypted using RSA Encryption: HELLO MY NAME IS
STEPHEN
Your un-encrpyted message is: [ HELLO MY NAME IS STEPHEN ]
Your encrypted message is: [ q8VLOqDvDGZoWTTZDzWMfVuIh6ayW0sdv0AkOQaEi7MFK
pUlMDdUtf9OwrOwmZUhYoXPg1ildUBt3d3Kei1MYhRlgGSgRZwmIIb+YATPPvmNKvsodyeKfmO
2mTECI/uNU9nz4YfZNcs5x/7ngv5fXXcZFYTdsDZJ6bGgubSAeyglyb1DQ1p8hpckJzRZnYsrR
2so/qbKZFJwm4vQN2RNCtanEjiitUfEPc1l6qNX5A1/xcweGMNwDsjVras/Sbm1xAl73T97BBv
GQvlQ1Sn1DRrSVsFkcDPYEdu1K3b34nvYZVSGU2GqiJEDpsOP18YO6moCnfmX922KIFdINOcMP
Q== ]
Your decrypted message is: [ HELLO MY NAME IS STEPHEN ]
MBP2697:RSA_Implementation stephen2697$
```