Important thing about RSA Asymmetric Encryption and Decryption

# Asymmetric encryption/decryption:

* everyone can encrypt (using a public key), only the host can decrypt (using a private key).

# Asymmetric signature/verification:

* only the host can sign (using a private key), everyone can verify (using a public key)

# The RSA (Rivest, Shamir, Adleman) encryption algorithm uses two Keys: Private and Public.

#### Scenario A

Suppose Alice wants to send a message to Bob (for his eyes only!). She can encrypt the message using the RSA algorithm with Bob's Public Key, which is not a secret (that's why they call it Public…). Once the message is encrypted, nobody can decrypt it, except the one holding the matching Private Key (that is Bob).

#### Scenario B

The reverse is also true: if Alice would encrypt the message using her own **Private Key**, Bob (and Eve, and everyone who can access this "encrypted" message) can decrypt it using Alice's Public Key. So, if everybody can decrypt it, what's the point in encrypting the message with a Private Key in the first place? Well, there is a point if Bob wants to make sure that the message has been written by Alice and not by someone else (Eve?).

# If really need a RSA Private Key Encryption, need to write a custom RSAEncryption Class

## **RSA Private Key Encryption**

Unfortunately, the RSACryptoServiceProvider class does not provide you this option, so I wrote my own implementation of the RSA algorithm using the basics of the RSACryptoServiceProvider in conjunction with Chew Keong TAN's class: BigInteger (<http://www.codeproject.com/KB/cs/biginteger.aspx>). At a low level, the RSA algorithm is about implementing mathematical equations on huge (huge) integers, so the BigInteger class is really essential. I couldn't have done it myself.

## **Using the RSAEncryption Class**

The class has six main methods:

Hide   Copy Code

void LoadPublicFromXml(string publicPath)

void LoadPrivateFromXml(string privatePath)

byte[] PrivateEncryption(byte[] data)

byte[] PublicEncryption(byte[] data)

byte[] PrivateDecryption(byte[] encryptedData)

byte[] PublicDecryption(byte[] encryptedData)

I believe the method names are self explanatory. First, you have to create a private / public key pair, using the .NET RSACryptoServiceProvider class. To do that, you just create an instance of this class and then call the appropriate methods, like this:

Hide   Copy Code

void LoadPublicFromXml(string publicPath)

RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();

File.WriteAllText(@"C:\privateKey.xml", rsa.ToXmlString(true)); *// Private Key*

File.WriteAllText(@"C:\publicKey.xml", rsa.ToXmlString(false)); *// Public Key*

*// Then, you can load those files to RSAEncryption instance:*

RSAEncryption myRsa = new RSAEncryption();

myRsa.LoadPrivateFromXml(@"C:\privateKey.xml");

myRsa.LoadPublicFromXml(@"C:\publicKey.xml");

*// Once the keys are loaded (if you load a private key, there is no need to*

*// load the public one) you can start Encrypt / Decrypt data*

*// using Private / Public keys.*

byte[] message = Encoding.UTF8.GetBytes("My secret message");

byte[] encryptMsg = myRsa.PrivateEncryption(message);

byte[] decryptMsg = myRsa.PublicDecryption(encryptMsg);

string originalMsg = Encoding.UTF8.GetString(decryptMsg);

*// returns "My secret message"*

.