Bouncy Castle - .Net Implementation - RSA Algorithm

In the earlier articles, we came across the concepts of RSA algorithm in Public Key cryptography. Being a generic topic, it can be implemented in security applications of variant technologies. We shall see the implementation of RSA algorithm using C#.

RSA Encryption

RSA Encryption can be achieved by following the below steps

1. Select an input message - For demonstration, we select to use a plain text message “Test message”
2. Generate a RSA key pair - The bouncy castle type – RSAKeyPairGenerator has been used for generating the asymmetric key pair. The RSAKeyPairGenerator uses two large prime numbers for generating the private and the public keys.
3. Create an RSA algorithm object - We need to create an object for the RSA asymmetric cipher.We can use the CipherUtilities collection of ciphers by specifying the exact padding and mode, or we may directly instantiate the algorithm.
4. Initialize the RSA algorithm for the encryption mode along with the asymmetric keys
5. Encrypt the message using the public key.
The below code snippet is written in C# language and makes use of Bouncy Castle APIs.

Code:

```
string inputMessage = "Test Message";
UTF8Encoding utf8enc = new UTF8Encoding();

// Converting the string message to byte array
byte[] inputBytes = utf8enc.GetBytes(inputMessage);

// RSAKeyPairGenerator generates the RSA Key pair based on the random number
and strength of key required
RsaKeyPairGenerator rsaKeyPairGnr = new RsaKeyPairGenerator();
rsaKeyPairGnr.Init(new Org.BouncyCastle.Crypto.KeyGenerationParameters(new
SecureRandom(), 512));
Org.BouncyCastle.Crypto.AsymmetricCipherKeyPair keyPair =
rsaKeyPairGnr.GenerateKeyPair();

// Extracting the public key from the pair
RsaKeyParameters publicKey = (RsaKeyParameters)keyPair.Public;

// Creating the RSA algorithm object
IAsymmetricBlockCipher cipher = new RsaEngine();
```

```
// Initializing the RSA object for Encryption with RSA public key. Remember,
for encryption, public key is needed
cipher.Init(true,publicKey);

//Encrypting the input bytes
byte[] cipheredBytes = cipher.ProcessBlock(inputBytes, 0,
inputMessage.Length);
```

RSA Decryption

Similarly, RSA Decryption can be done through below steps -

1. Input the encrypted message.
2. Select the same RSA algorithm.
3. Decrypt the message using the private key.
4. Compare the decrypted with original message.

Code:

```
// Extracting the private key from the pair
RsaKeyParameters privateKey = (RsaKeyParameters)keyPair.Private;
cipher.Init(false, privateKey);
byte[] deciphered = cipher.ProcessBlock(cipheredBytes, 0,
cipheredBytes.Length);
string decipheredText = utf8enc.GetString(deciphered);
```

On comparing decipheredText with inputMessage, we would find both of them to be equal.

Bouncy Castle - .Net Implementation of RSA Algorithm with OAEP padding

RSA algorithm has been found to be weak because it has no random component. An attacker might create a database of possible input messages and the encrypted text given by the RSA algorithm using the same public key. Then, he would simply compare the two encrypted messages and would know the original message. To avoid this possibility, we might like to use Padding schemes. One of such padding scheme is OAEP Optimal Asymmetric Encryption Padding (OAEP). It adds a factor of randomness which makes it impossible to determine the original plain text.

Encryption steps-

1. Select an input message - For demonstration, we select to use a plain text message “Test message” and generate a RSA key pair.
2. Create an RSA algorithm object.
3. Create a cipher object for OAEP encoding
4. Initialize the RSA algorithm for the encryption mode along with the asymmetric keys specifying the hash digest that will be used for calculating the randomness.
5. Process the blocks and encrypt the message using the public key.

   Code:

```
// Encryption steps ----------------------------------
SHA256Managed hash = new SHA256Managed();
SecureRandom randomNumber = new SecureRandom();
byte[] encodingParam =
hash.ComputeHash(Encoding.UTF8.GetBytes(randomNumber.ToString()));
string inputMessage = "Test Message";
UTF8Encoding utf8enc = new UTF8Encoding();

// Converting the string message to byte array
byte[] inputBytes = utf8enc.GetBytes(inputMessage);

// RSAKeyPairGenerator generates the RSA Key pair based on the random number
and strength of key required
RsaKeyPairGenerator rsaKeyPairGnr = new RsaKeyPairGenerator();
rsaKeyPairGnr.Init(new Org.BouncyCastle.Crypto.KeyGenerationParameters(new
SecureRandom(), 1024));
Org.BouncyCastle.Crypto.AsymmetricCipherKeyPair keyPair =
rsaKeyPairGnr.GenerateKeyPair();
RsaKeyParameters publicKey = (RsaKeyParameters)keyPair.Public;
RsaKeyParameters privateKey = (RsaKeyParameters)keyPair.Private;
IAsymmetricBlockCipher cipher = new OaepEncoding(new RsaEngine(), new
Sha256Digest(), encodingParam);
cipher.Init(true, publicKey);
byte[] ciphered = cipher.ProcessBlock(inputBytes, 0, inputMessage.Length);
string cipheredText = utf8enc.GetString(ciphered);

// Decryption steps ------------------------------------------
cipher.Init(false, privateKey);
byte[] deciphered = cipher.ProcessBlock(ciphered, 0, ciphered.Length);
string decipheredText = utf8enc.GetString(deciphered);
//----------------------------------------------------------------
```

Source: http://www.go4expert.com/articles/bouncy-castle-net-implementation-rsa-t24827/