Lab 8: Public-Key Encryption

Details

Aim: To provide a foundation in asymmetric encryption, using the RSA method.

Activities

1. .NET provides us with an excellent foundation in creating applications in which we can view and log events, as well as monitoring for processes. Another key feature is that it supports many encryption and authentication standards. If Visual Studio is installed on your machine, download the following solution:

↳http://www.dcs.napier.ac.uk/~bill/eventLogNew.zip

It has a Windows interface, such as:

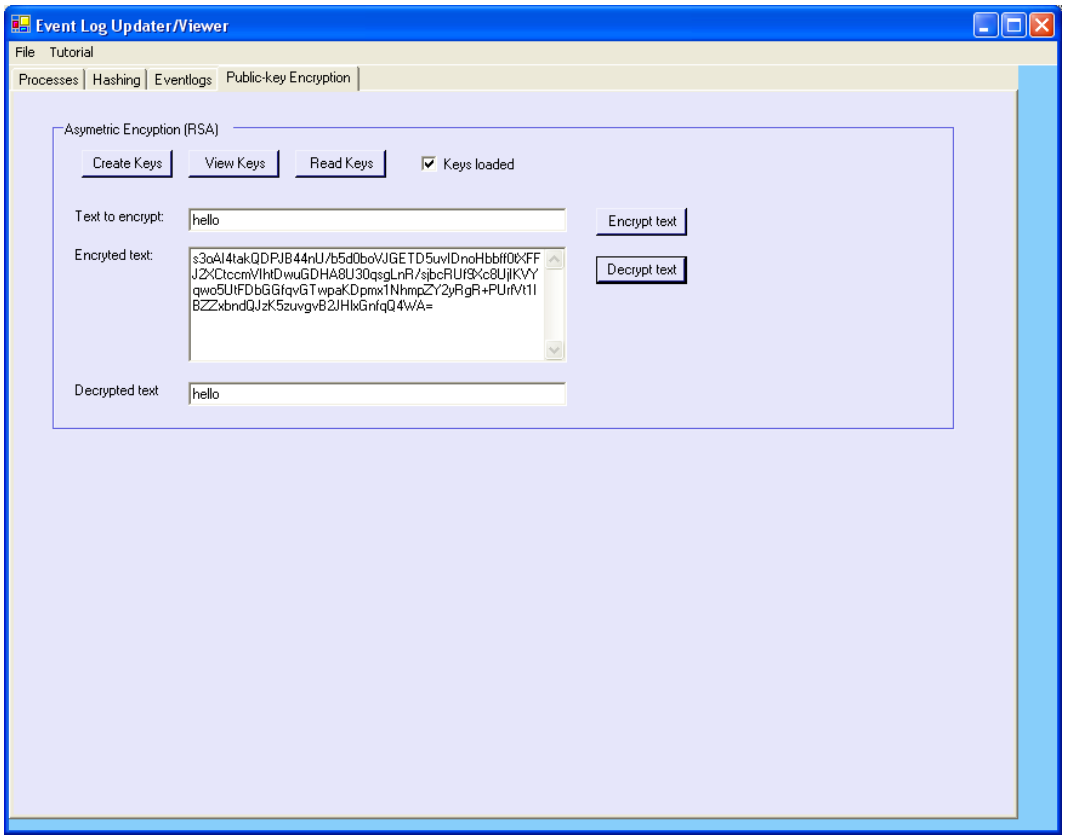


Figure 1: Public-key encryption

2. For the **Create Keys** button add the following code:

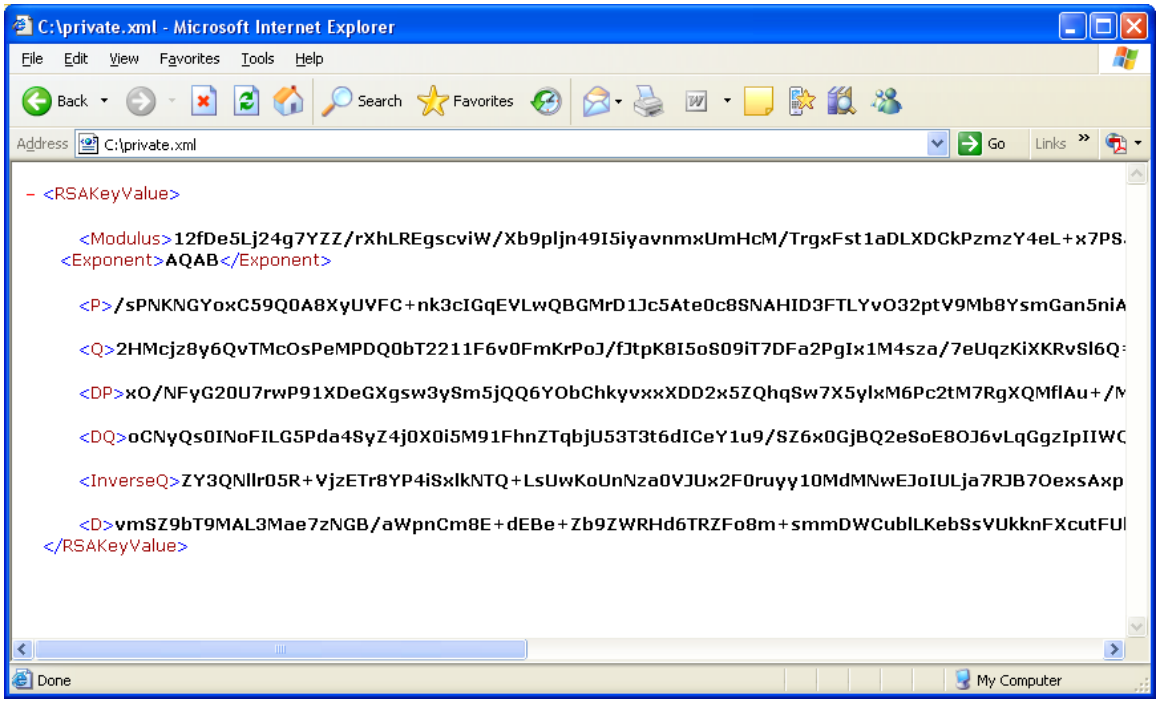
```
System.Security.Cryptography.RSACryptoServiceProvider RSAProvider;
RSAProvider = new System.Security.Cryptography.RSACryptoServiceProvider(1024);
publicAndPrivateKeys = RSAProvider.ToXmlString(true );
justPublicKey = RSAProvider.ToXmlString( false);
StreamWriter fs = new StreamWriter("c:\\public.xml");
fs.Write(justPublicKey);
fs.Close();
fs = new StreamWriter("c:\\private.xml");
fs.Write(publicAndPrivateKeys);
fs.Close();
checkBox2.Checked=true;
```

3. This creates two files on your disk. One contains your public key (**public.xml**) and the other contains both the private key and the public key (**private.xml**). Run the program, and using the View Keys button, view the keys.

What is the format of the keys:

View the files using Internet Explorer to see the XML format.

What are the XML tags in each of the files:



4. From the form, add the following code to the **Read Keys** button:

```
XmlTextReader xtr = new XmlTextReader("c:\\private.xml");
publicAndPrivateKeys=""; // reset keys
justPublicKey="";
while (xtr.Read())
{
   publicAndPrivateKeys += xtr.ReadOuterXml();
}
xtr.Close();
xtr = new XmlTextReader("c:\\public.xml");
while (xtr.Read())
{
   justPublicKey += xtr.ReadOuterXml();
}
xtr.Close();
checkBox2.Checked=true;
```

5. Now add the following code to the **Encrypt text** button:

```
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
string txt=tbTxtEncrypt.Text;
rsa.FromXmlString(justPublicKey);
byte[] plainbytes = System.Text.Encoding.UTF8.GetBytes(txt);
byte[] cipherbytes = rsa.Encrypt(plainbytes,false);
this.tbTxtEncrypted.Text=Convert.ToBase64String(cipherbytes);
```

6. Now add the following code to the **Decrypt text** button:

```
RSACryptoServiceProvider rsa = new RSACryptoServiceProvider();
string txt=tbTxtEncrypted.Text;
rsa.FromXmlString(publicAndPrivateKeys);
byte[] cipherbytes = Convert.FromBase64String(txt);
byte[] plainbytes = rsa.Decrypt(cipherbytes,false);
System.Text.ASCIIEncoding enc = new System.Text.ASCIIEncoding();
this.tbTxtDecrypt.Text = enc.GetString(plainbytes);
```

7. Now run the program and add some text to the Text to encrypt box, and see if the program encrypts the text, and correctly decrypts it.

Did the program encrypt and decrypt correctly:

8. Now get your give your neighhour your public key file (**public.key**), and get them to encrypt a message. Now take the encrypted message (pass it through copy and paste, and then email the cipertext, or put it on a shared folder), and see if can decrypt it.

Did the program decrypt correctly: