

Re: RSACryptoServiceProvider decrypt with public key

Source: <http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.security/2004-05/0315.html>

From: Martin M?ller (*mav.northwind_at_web.de*)

Date: 05/28/04

Date: 28 May 2004 07:53:44 -0700

Thanks for shedding some light on the topic.

In fact I was under the impression that both ways should be possible.

My intention was to create some encrypted license information with my private key which my programs could decipher using a public key I've embedded in the program.

This way the license information would be secret for everybody without the matching public key and could be decoded by my program with the embedded public key. Of course you could somehow get the public key from the program, but secrecy was in fact not the main goal of this approach.

I wanted to be sure that the license data was authentic, and since you can't encode data with a public key and then decode with the same public key you couldn't create your own license information.

But since private key encryption and public key decryption isn't possible I've switched to keeping the data as plaintext and just signing it so that I can be sure it has not been tampered with.

This works just fine....

Best regards,

Mav

"Stephen McCloskey [MSFT]" <stemccl@online.microsoft.com> wrote in message news:<etatvQfQEHA.2456@TK2MSFTNGP10.phx.gbl>...

> *Hello Martin,*

>

>

>

> *You are correct that it is possible to both encrypt with the public*

> *key/decrypt with the private key and encrypt with the private key/decrypt*

> *with the public key. However, it's a little more complicate than just that.*

> *Here's why:*

>

>

microsoft.public.dotnet.security: Re: RSACryptoServiceProvider decrypt with public key

- >
- > *Your public key is meant for public consumption by anyone. Your private key*
- > *must always stay private and must only be known by you.*
- >
- >
- >
- > *This basic scheme offers the following scenarios:*
- >
- >
- >
- > *If Alice gives Bob her public key (by sending it across the wire), Bob can*
- > *encrypt data and send it back to Alice. Because only Alice has the private*
- > *key, only she can decrypt Bob's data. Now, let's assume that a third party*
- > *named Eve is able to listen in to the conversation. Eve would be able to*
- > *see the public key and the encrypted data, but she could not decrypt Bob's*
- > *data and listen in. So, the scenario where Bob encrypts data and sends it*
- > *to Alice using her public key is a useful way for Bob to share his data*
- > *without Eve being able to listen in.*
- >
- >
- >
- > *Now let's look at the scenario where Alice encrypts data with her private*
- > *key and sends it to Bob. In this situation Alice sends her public key to*
- > *Bob and then sends some data to Bob. Bob then uses the public key to*
- > *decrypt the data. However, let's assume that Eve is still lurking out there*
- > *and that she is able to see both the public key and the encrypted data. In*
- > *this situation, Eve has everything she needs to decrypt Alice's data. So,*
- > *as you can see, when Alice encrypts data with her private key, there isn't a*
- > *way for her to keep the data secret. In fact this scheme buys you*
- > *absolutely nothing if your goal is keeping data secret! If Alice wants to*
- > *send Bob a secret using asymmetric cryptography, she must use Bob's public*
- > *key (not her private key).*
- >
- >
- >
- > *But . there is one thing that this second scheme does buy Alice and Bob.*
- > *When Alice encrypts data with her private key and sends it to Bob along with*
- > *her public key, Bob can verify that the data came from Alice. As long as*
- > *Bob trusts that only Alice has access to the private key, he knows that data*
- > *that will decrypt with Alice's public key actually came from Alice, and not*
- > *from say .Eve. So, imagine if Alice sent Bob a message and then sent a*
- > *hashed version of the message that was encrypted with her private key. Bob*
- > *could then decrypt the hash and compare it to a hash of the plaintext*
- > *version that he generates. If both hashes match, then Bob knows that Eve*
- > *did not hijack the message and send him a false one. He knows that this is*
- > *an unaltered message that only came from Alice. This is how digital*
- > *signatures work!*
- >
- >
- >
- > *Now, let's get back to the RSACryptoServiceProvider class. The Encrypt*

microsoft.public.dotnet.security: Re: RSACryptoServiceProvider decrypt with public key

- > *method ONLY encrypts using the public key and the Decrypt method only*
- > *decrypts using the private key. This makes sense because we have already*
- > *established that encryption with a private key for the purpose of keeping*
- > *data secret is quite pointless.*
- >
- >
- >
- > *The RSACryptoServiceProvider.ToXmlString method will return a public key*
- > *only if you pass "false". It will return both a public and private key when*
- > *you pass "true". So, when you try to decrypt data using the*
- > *RSACryptoServiceProvider.Decrypt method, but you only initialize the*
- > *RSACryptoServiceProvider using an XML file with a public key, you will*
- > *always get an exception because there is no private key to perform*
- > *decryption. This is the expected behavior, but hopefully it makes more*
- > *sense now.*
- >
- >
- >
- > *You can use the RSACryptoServiceProvide class to create a digital signature*
- > *using methods like SignData, SignHash, VerifyData, and VerifyHash. These*
- > *are the methods that use the private key to encrypt a value (a hash) and a*
- > *public key to verify the value, but they are designed to be used only for*
- > *digital signatures. They can't be used to encrypt arbitrary data with a*
- > *private key and decrypt the data with a public key.*
- >
- >
- >
- > *As always, there are lots of caveats to cryptography, so the real story is*
- > *slightly more complicated than what I have presented. However, I hope this*
- > *clears things up. Let me know if you have any more questions.*
- >
- >
- >
- > *Stephen McCloskey*
- >
- > *Programmer/Writer*
- >
- > *.NET Framework SDK*
- >
- >
- > *"Martin M?ller" <mav.northwind@web.de> wrote in message*
- > *news:e81857a.0405190403.2fab6118@posting.google.com...*
- > > *Dear community, please help:*
- > > *For several days now I've been trying to implement something that*
- > > *should be possible according to all the sources you find on asymmetric*
- > > *encryption but just can't get it to work.*
- > >
- > > *The main idea behind asymmetric encryption is that there's a public*
- > > *and a private key and that the public key can be derived from the*
- > > *private key, but not the other way round.*
- > > *Data encrypted with the public key can be decrypted using the private*

microsoft.public.dotnet.security: Re: RSACryptoServiceProvider decrypt with public key

> > key.
> > *Data encrypted with the private key can be decrypted using the public*
> > *key.*
> >
> > *The first way (encrypt with public, decrypt with private) is shown in*
> > *several examples you can find, but the other one (encrypt with*
> > *private, decrypt with public) doesn't seem to work and I can't find a*
> > *working example for it either.*
> >
> > *I'm using .NET 1.0 on a WinXP machine and tried the following:*
> > *Create a new RSACryptoServiceProvider, save each key to a separate*
> > *file using ToXmlString().*
> >
> > *For encryption I read the private key file, use a new*
> > *RSACryptoServiceProvider's FromXmlString() method to set the*
> > *parameters just read and encode the data to a Base64 string.*
> > *The data encrypted is very short (about 20 bytes) and key length is*
> > *sufficient (1024, but you get the same result with other key lengths).*
> > *Encryption works fine.*
> >
> > *Now I create a new CSP like before, read the public key file, convert*
> > *the Base64 string to a byte array again and try to decrypt it and I*
> > *always get a CryptographicException stating "invalid key", if I don't*
> > *use OAEP padding and "Error occurred while decoding OAEP padding."*
> > *when using padding.*
> >
> > *The other way (encrypt with public and decrypt with private) does*
> > *work, as does encryption and decryption both with the private key, but*
> > *if I always need the private key what's the point of asymmetric*
> > *encryption ?*
> >
> > *Can anyone enlighten me? Is it a bug in .NET 1.0? Does 1.1 behave*
> > *differently? What other options do I have?*
> >
> >
> > *Here are the code fragments I described above:*
> > ----- Create a key pair -----
> > *RSACryptoServiceProvider csp = new RSACryptoServiceProvider(1024);*
> >
> > *string s = csp.ToXmlString(true);*
> > *StreamWriter sw = new StreamWriter("KeyPriv.xml");*
> > *sw.WriteLine(s);*
> > *sw.Close();*
> >
> > *s = csp.ToXmlString(false);*
> > *sw = new StreamWriter("KeyPub.xml");*
> > *sw.WriteLine(s);*
> > *sw.Close();*
> >
> > *csp.Clear();*
> >

```
>> ----- Encrypt with private key -----
>> RSACryptoServiceProvider csp = new RSACryptoServiceProvider(1024);
>>
>> StreamReader sr = new StreamReader("KeyPriv.xml");
>> string s = sr.ReadToEnd();
>> sr.Close();
>> csp.FromXmlString(s);
>>
>> byte[] inp = System.Text.Encoding.Unicode.GetBytes(clearText);
>> byte[] outp = csp.Encrypt(inp, false);
>> cypherText = Convert.ToBase64String(outp);
>> csp.Clear();
>>
```

```
>> ----- Decrypt with public key -----
>> RSACryptoServiceProvider csp = new RSACryptoServiceProvider(1024);
>>
>> StreamReader sr = new StreamReader("KeyPub.xml");
>> string s = sr.ReadToEnd();
>> sr.Close();
>> csp.FromXmlString(s);
>>
>> byte[] inp = Convert.FromBase64String(cypherText);
>> // This always throws an exception "invalid key" :(((
>> byte[] outp = csp.Decrypt(inp, false);
>> clearText = System.Text.Encoding.Unicode.GetString(outp);
>> csp.Clear();
>>
```
