

## Re: Using the Public Key embedded in the Assembly?

**Source:** <http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.security/2004-10/0023.html>

---

**From:** roland ([roland.demeester\\_at\\_skynet.be](mailto:roland.demeester_at_skynet.be))

**Date:** 10/02/04

Date: Sat, 2 Oct 2004 15:56:20 +0200

Shawn

Thanks for your reply.

I am not sure if I follow you.

I have put the RSA keypair that I created via sn.exe on my machine in a container: I used sn.exe -i [keyfile] [containername]. This made it possible to create an RSACryptoProvider in a program on my machine to sign a license file using my private key. The .snk file is used by the compiler to sign the assembly and to embed the public key in the assembly.

When I send the assembly to a customer (together with the license file), the assembly is only containing the public key, not the private key (I hope so!) and no csp container with such name and content exists on the machine of the customer. So to verify the signature, I need to transform the byte array I get from [AssemblyName.GetPublicKey] in a useable form.

I don't understand how the StrongNameKeyInstall API can help me on the machine of the customer: doesn't it need the .snk file (i.e. the keypair)?

Regards, Roland

""Shawn Farkas [MS]"" <[shawnfa@online.microsoft.com](mailto:shawnfa@online.microsoft.com)> wrote in message news:W64KMDAqEHA.3944@cpmsftngxa06.phx.gbl...

> *Hi Roland,*

>

> *Instead of parsing the blob yourself, I would instead try to put it in a*  
> *key container, and then use the RSACryptoServiceProvider constructor that*  
> *reads keys out of containers. If you're on Whidbey, you can use the new*  
> *ImportCspBlob method on RSACryptoServiceProvider, but if you're on Everett*  
> *you'll have to do a little more work.*

> *You can use the exposed StrongNameKeyInstall API. You can find:*

> *\* some details on using this API here:*

> <http://blogs.msdn.com/shawnfa/archive/2004/10/01/236773.aspx>

> *\* the P/Invoke declaration for it here:*

> <http://blogs.msdn.com/shawnfa/articles/236725.aspx>

> *\* and a sample of using it here;*

> <http://blogs.msdn.com/shawnfa/articles/236731.aspx>

> *Specifically, you'll want to look at the InstallKey method on line 149 of*  
> *that last link.*

>

microsoft.public.dotnet.security: Re: Using the Public Key embedded in the Assembly?

- > *Once you've gotten a key into the key container, you can create a*
- > *CspParameters object that specifies the key container name you installed*
  
- > *to, and pass that object to the RSACryptoServiceProvider constructor.*
- >
- > *When you've finished your validation, the same links above will give a*
- > *sample of the StrongNameKeyDelete API, which will remove the key*
- > *container.*
- >
- > *-Shawn*
- > <http://blogs.msdn.com/shawnfa>
- > --
- > *This posting is provided "AS IS" with no warranties, and confers no*
- > *rights.*
- >
- >
- > *Note:*
- > *For the benefit of the community-at-large, all responses to this message*
- > *are best directed to the newsgroup/thread from which they originated.*
- > -----
- > *From: "roland" <roland.demeester@skynet.be>*
- > *Subject: Using the Public Key embedded in the Assembly?*
- > *Date: Fri, 1 Oct 2004 09:21:25 +0200*
- > *Lines: 48*
- > *X-Priority: 3*
- > *X-MSMail-Priority: Normal*
- > *X-Newsreader: Microsoft Outlook Express 6.00.2800.1437*
- > *X-MimeOLE: Produced By Microsoft MimeOLE V6.00.2800.1441*
- > *Message-ID: <#T1kkd4pEHA.3252@TK2MSFTNGP14.phx.gbl>*
- > *Newsgroups: microsoft.public.dotnet.security*
- > *NNTP-Posting-Host: 46.180-201-80.adsl.skynet.be 80.201.180.46*
- > *Path: cpmsftngxa06.phx.gbl!TK2MSFTNGP08.phx.gbl!TK2MSFTNGP14.phx.gbl*
- > *Xref: cpmsftngxa06.phx.gbl microsoft.public.dotnet.security:7590*
- > *X-Tomcat-NG: microsoft.public.dotnet.security*
- >
- > *L.S.*
- > *I want to build-in a license scheme in my controls.*
- > *The concept is to have the public key embedded in my assembly; a*
- > *licenseprovider then retrieves this public key and uses it to verify the*
- > *signature of the license file. The license file is unique to each*
- > *licensee,*
- > *so if the license file is going astray, I always can trace the source.*
- >
- > *This is how I implemented this:*
- >
- > *I used sn.exe to create an RSA keypair that I refer to in my assembly*
- > *and*
- > *I*
- > *stored this key pair (via sn.exe -i) in a named csp container. This*
- > *embeds*
- > *the public key in my assembly. In my license file creation program I use*

microsoft.public.dotnet.security: Re: Using the Public Key embedded in the Assembly?

> an  
>> *RSACryptographicProvider based on cspParameters from this named container.*  
>> *During execution I retrieve the public key from the assembly through*  
>> *[Assembly].GetExecutingAssembly().getName.getPublicKey. This gives me a*  
> byte  
>> *array, 160 long. The problem is that the methods for verifying the*  
> signature  
>> *in a signedXML document are using a RSACryptographicProvider and not*  
this  
>> *publicKey as a byte array. By browsing the user groups I found (was*  
> 'told')>> *that I can retrieve the modulus and the exponent from this byte array:*  
the  
>> *exponent should be equal to the last 3 elements and the modulus should*  
be  
>> *128 elements long and starting at 27th element.*  
>> *This should make it possible to create such a provider and use it to*  
> verify  
>> *the signature.*  
>> *'Create a new instance of RSACryptoServiceProvider.*  
>> *Dim \_rsa As RSACryptoServiceProvider = New RSACryptoServiceProvider*  
>> *Dim \_RSAKeyInfo As RSAParameters = New RSAParameters*  
>> *'Set \_RSAKeyInfo to the public key values.*  
>> *\_RSAKeyInfo.Modulus = \_modulus '(a byte array extracted from the*  
> publickey  
>> *array)*  
>> *\_RSAKeyInfo.Exponent = \_exponent '(idem)*  
>> *'Import key parameters into the provider.*  
>> *\_rsa.ImportParameters(\_RSAKeyInfo)*  
>> *...*  
>> *return signedXml.CheckSignature(\_rsa)*  
>> *But this doesn't work!*  
>> *When I extract the public key by using ToXMLString(False) in both cases,*  
I  
>> *get a totally different result for the public key: the modulus of the*  
> public  
>> *key retrieved from the csp container is only some 88 characters long,*  
> while  
>> *the one retrieved from the embedded public key in the assembly is some*  
160  
>> *characters long. Also the exponents are totally different (although*  
their  
>> *length is the same: 3).*  
>>  
>> *Obviously I am doing something wrong. Can anybody point me to the*  
> solution?  
>> *Thanks in advance.*  
>>  
>> *Roland*  
>>

microsoft.public.dotnet.security: Re: Using the Public Key embedded in the Assembly?

> >

> >

>