

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

**Source:** <http://www.derkeiler.com/Newsgroups/microsoft.public.dotnet.security/2004-10/0057.html>

---

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

**Date:** 10/05/04

Date: Tue, 5 Oct 2004 21:19:49 +0200

Nicole

I read your comments and started some investigations. I finally understand the Keyblob, thanks to Michel Gallant ([www.jensign.com](http://www.jensign.com)), who wrote a lot of free cryptographic utilities, amongst them a keyblob parser. I found out as well that the keypair I retrieve from the container is not the same as the one in the keyfile created by sn.exe, although the container is created with sn.exe -i. I am now retrieving the keypair for signing straight from the keyfile. Everything seems to work well now, but I am currently streamlining the code and will publish it shortly to GotDotNet. Thanks very much for your help. Let me know if you want some details in the meantime.

Greetings, Roland

"Nicole Calinoiu" <[ngcalinoiu\\_REMOVETHIS\\_AT\\_gmail\\_DOT\\_com](mailto:ngcalinoiu_REMOVETHIS_AT_gmail_DOT_com)> wrote in message news:%23x651thqEHA.3744@TK2MSFTNGP10.phx.gbl...

- > *Sorry, late Friday brain fart--I was focused on the mismatch, not what you*
- > *were attempting to do with the keys. At any rate, the first problem to*
- > *solve is still to ensure compatible formatting of the keys. If whatever*
- > *approach you've been trying to get the private key out of the CSP is not*
- > *giving you a result that is in an identical format as that offered by the*
- > *.NET Framework for an assembly's public key, you need to fix that problem*
- > *before attempting to sign your file.*
- >
- > *Instead of trying to map at verification, why not fix the original reading*
- > *problem if there is one? To test this, read the \_public\_ key out of the*
- CSP
- > *rather than the private key. If the result is not the same as the public*
- > *key read from the assembly, then you know that you need either an*
- alternate
- > *approach for reading from the CSP or some post-read massaging. If you*
- need
- > *help with this, perhaps you could let us know how you're reading the*
- private
- > *key in the first place.*
- >
- > *If the result is already the same, perhaps the problem lies with your*
- > *signing code rather than the verification code. Since you've only*

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

provided

> *the verification side, it's a wee bit difficult for other folks to try to  
> reproduce the problem without some additional details...*

>

>

>

> *"roland" <roland.demeester@skynet.be> wrote in message*

> *news:uwqshSIqEHA.556@TK2MSFTNGP11.phx.gbl...*

>> *Nicole,*

>> *Thanks for your reply.*

>> *However, your idea is not workable: you are not signing the license file*

>> *(or*

>> *the digest) with your public key, but with the private key. The assembly*

>> *is*

>> *then verifying this signature with the (embedded) public key. So the*

>> *public*

>> *key in the assembly is of no use at that time. The issue is to retrieve*

>> *the*

>> *public key in the assembly, by the assembly (for instance in design  
mode),*

>> *to verify the signature of the license and this happens on the machine  
of*

>> *the licensee, where no csp container is created before.*

>> *Regards,*

>> *Roland*

>>

>>

>> *"Nicole Calinoiu" <ngcalinoiu REMOVETHIS AT gmail DOT com> wrote in*

>> *message*

>> *news:OriTVN7pEHA.2864@TK2MSFTNGP12.phx.gbl...*

>>> *Roland,*

>>>

>>> *Why are you using a different source (CSP vs assembly signature) for  
the*

>>> *license signing and verification? Why not read the public key from  
your*

>>> *control assembly (of which I'm guessing you have a copy <g>) when  
signing*

>>> *the license file? This would at least give you a consistent value for*

>>> *the*

>>> *key, so any remaining discrepancies would like with your signing and/or*

>>> *verification code instead of the data.*

>>>

>>> *HTH,*

>>> *Nicole*

>>>

>>>

>>> *"roland" <roland.demeester@skynet.be> wrote in message*

>>> *news:%23T1k4d4pEHA.3252@TK2MSFTNGP14.phx.gbl...*

>>>> *L.S.*

>>>> *I want to build-in a license scheme in my controls.*

Re: Using the Public Key embedded in the Assembly?

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

> > > *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  
> > > 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)  
> > > ...**

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

> >> > *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*  
> >> >  
> >> >  
> >>  
> >>  
> >  
> >  
>  
>  
>