

## Re: How good an encryption algorithm is this?

**Source:** <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2004-11/1040.html>

---

**From:** Ian Griffiths [C# MVP] (*ian-interact-sw\_at\_nospam.nospam*)

**Date:** 11/25/04

Date: Thu, 25 Nov 2004 16:22:08 -0000

"Bonj" wrote:

>> *For example, if I want to send data from my laptop to my server at home,*  
>> *then it's relatively straightforward for me to make sure that the key is*  
>> *not*  
>> *known by anyone other than those two computers.*  
>  
> *No offence, and that may be a very interesting and productive thing*  
> *for you to do, but how is it relevant to this thread?*

I came up with the example because you hadn't previously told us your full requirements. (I know you think you had, but it was only clear from your most recent problem statement.) Clear the example is totally relevant to the question in the subject line. As it happens, it's also relevant to your problem, despite what you appear to think.

Now that I understand your requirements, I am confident that what I described is very relevant because it illustrates a weakness in your algorithm, one that can be exploited in the scenario you describe as well as in the scenario I described.

The weakness described in the laptop <-> home server scenario is the same as the weakness in your scenario: anyone who gets a way of seeing the encrypted data will be able to use a statistical attack to work out what the key is. The only difference between the scenario I described and the one you describe is where this encrypted data lives, and by extension, the way in which the attacker gets her hands on that data. In the networked scenario, you'd use a packet sniffer. But in your case, you would need to harvest encrypted passwords from the registries of various users' machines. But having obtained sufficient encrypted data, the statistical crypto attack is identical in both cases because the attack is the same in both cases.

The fact that I'm doing this with data in the registry rather than data being sent over the network is wholly irrelevant – the important thing about the example I gave is that the encrypted data that is directly visible to an attacker. (You've made it clear this is the case, since the attack you describe involves the attacker having access to the machine. They would need to get this access repeatedly to launch this particular attack of

## microsoft.public.vc.language: Re: How good an encryption algorithm is this?

course.) And even if the key may not be as directly visible to them, the encrypted data can be used to discover the key. (Or a bit pattern that is as good as the key.)

It's the fact that your algorithm makes it fairly easy to deduce the key given nothing but the encrypted data that makes it a bad algorithm.

Things get even easier if the attacker is able to run your program – your algorithm then makes it supremely easy to write the 'crack program' that you are worried about. All I need to do is make up a nice long password, and get your program to store it. Then I can XOR my credentials against what you stored in the database, and I've got your XOR bit pattern. I can now use this to recover anyone's credentials from what is stored in their registry. (This is a 'chosen plaintext' attack.)

> *This is a case of where to store the key on the client. As far as the encryption goes, it's nothing to do with encrypting it on one machine and decrypting it again on the other*

Actually that's precisely what it might be. The attack would look like this: the attacker gradually acquires the encrypted data in the registry from her colleagues' machines. She builds up a collection of such pieces of data, and when she has enough to succeed in her statistical attack, she is now able to decrypt the data on her own machine.

The important thing is not whether the data went over a network (it probably didn't with this attack – a USB disk would be the obvious mode of transfer). Nor is it relevant whether or not the data got transferred. It's the fact that it is visible to the attacker. After all, if the data was not visible to your attacker, why on earth would you bother encrypting it? Isn't the fact that attackers will be able to see the data the whole reason you want to encrypt? So the fact that your algorithm is useless if the attacker can see the data is the problem.

> *– I don't know why everybody instantly assumes that it is – maybe because it makes keeping the key secret easier?*

The reason people keep coming up with networked examples is not because they think that this is what's happening in your case. It's because they are assuming (correctly) that in your scenario, encrypted data will be visible to an attacker. So they are talking about a very common scenario in which encrypted data becomes visible to the attacker. The way in which the data becomes visible to the attacker is different, but the way in which it becomes visible isn't relevant.

The fact that a network is involved in these examples isn't the point. The key feature of all these examples is that the encrypted data is visible to the attacker, just like it is in your scenario. That's why these examples are all wholly relevant to your scenario. People aren't assuming that your data is being passed over the network, they're just showing you common and

Re: How good an encryption algorithm is this?

microsoft.public.vc.language: Re: How good an encryption algorithm is this?

well-understood scenarios \*in which the data is visible to the attacker\*.  
The data is visible to the attacker in your scenario too, which is why you should be paying attention to these examples.

>> *Fortunately I'm not using your algorithm. So my key has a good  
>> chance of remaining secret.*  
>  
> *Great – I'm happy for you. But how does \*mine\* stand a chance of  
> remaining secret, when it is a problem of persistence rather than one  
> of transfer?*

Are you saying that the encrypted data will not be visible to the attacker?  
The issue of transfer is irrelevant, so please stop getting hung up on it.  
The only relevant issue is what's visible to the attacker and what's not.

As for how your key will remain secret... Well that's why you use the DPAPI. The DPAPI makes it very much more difficult to discover what the key is. The key is not stored directly anywhere on the system – it is derived from a number of different sources, some of which are not accessible to normal users. (If your attacker is prepared to remove the hard disk, clone it, and perform forensics on it to discover these hidden keys, then all bets are off. But if your attacker is that determined, then you're probably out of luck. Theoretically the attacker could also write a device driver to do the same thing, but it's a very difficult attack; I've never heard of anyone executing it successfully. And again it requires a level of skill and determination that's quite formidable. And it would only be an option if the machine was left unattended while logged in with an admin account.)

>> *I'm still not quite sure exactly what it is you're trying to achieve  
>> here.*  
>  
> *Read the full thread. I've explained it many times, and posted a link to  
> someone else's similar scenario.*

I've read the full thread several times and with all due respect, as far as I can tell you've only just recently posted a full coherent explanation. (And I don't think I'm alone in thinking that, as everyone seemed to be confused as to what you're trying to achieve until just lately.)

And don't forget that you started by asking "How good an encryption algorithm is this?" which is a much more general question. You can't blame people for answering the question you put in the subject line!

>> *and the DPAPI, which is a technology built into Windows that is  
>> designed to solve pretty much exactly this problem.)*  
>  
> *That's probably what I'll go with. There's still unanswered questions  
> though, such as what if two users just happen to have the same password.*

The DPAPI uses various input data to drive the way in which it encrypts data. If two different users encrypt \*exactly\* the same data, it will end

Re: How good an encryption algorithm is this?

microsoft.public.vc.language: Re: How good an encryption algorithm is this?

up storing different data, because it uses user identity as one of the inputs to the key generation. (Actually it's better than that – merely knowing who the user was isn't sufficient – you need to be logged in as that user to decrypt the data.) Also, if the same user stores the same secret on two different machines, the data stored is also different, because DPAPI uses the machine identity as one of the inputs to its encryption algorithm.

--

Ian Griffiths - <http://www.interact-sw.co.uk/ianblog/>  
DevelopMentor - <http://www.develop.com/>