

microsoft.public.dotnet.languages.csharp: Re: How good an encryption algorithm is this?

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

**Source:**

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-11/5698.html>

---

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

**Date:** 11/25/04

Date: Thu, 25 Nov 2004 16:23:17 -0000

Actually it's vitally important that the salt is different every time.

And the salt is not secret. It's only there to make dictionary attacks harder, by ensuring that identical data hashes to different values. It's not there to enable decryption.

--

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

DevelopMentor - <http://www.develop.com/>

"Bonj" <[Bonj@discussions.microsoft.com](mailto:Bonj@discussions.microsoft.com)> wrote in message news:E4AD84A3-26E8-456B-B175-2D2E2C72BB9E@microsoft.com...

> OK thanks

> Just one question - presumably the "salt" always has to be the same?

> If so, does it not suffer from the same problems of secret persistence as the key itself does?

>

>

> "Igor Tandetnik" wrote:

>

>> "RoyFine" <[rlfine@obfuscate.net](mailto:rlfine@obfuscate.net)> wrote in message

>> news:e0jddxj0EHA.3596@TK2MSFTNGP12.phx.gbl

>> > consider a database table that has encrypted passwords (not simple xor

>> > mapping encryption, but a one-way hash of the password). if i look

>> > at the encrypted password values, it seems like just so much muddle

>> > to me. but in my spare time, i hash 5 million or so common passwords

>> > (in prior spare time, i wrote a program to generate these) and i save

>> > the hashed value with the plaintext. then i look in your password

>> > table, and i just might find a few values there that are the same as

>> > the ones that i computed - if we used the same algorithm to hash,

>> > then i have just discovered a few passwords. this is a dictionary

>> > attack (using my dictionary of plaintext trial passwords and the

>> > corresponding hash)! the literature suggests that with todays

>> > computer power, these sorts of attacks are trivial and you can break

>> > an entire password file of a hundred or so in just minutes.

>> >

>> > Enter \*Salt\* - salt is a random string that is concatenated with the

>> > plaintext passwod before you run it through the hash (one-way)

>> > function, then both the salt and the one way has are stored in the

>> > database. if you are using a system generated guid, then every

>> > stored value is now 128 bits longer. but the dictionary attack just

>> > got a lot harder - now i have to compute the dictionary once for

>> > every password/salt combination. now, instead of minutes to recover

>> > the passwords, the time jumps up to a couple of weeks - see feldmeier

>> > and karn, unix security-10 years later, applied cryptography [pg

Re: How good an encryption algorithm is this?

microsoft.public.dotnet.languages.csharp: Re: How good an encryption algorithm is this?

```
>> > 52-53] by bruce scheiner, and the following link:
>> > http://groups.google.com/groups?selm=690j3h%24115%40bqtnsc02.worldnet.att.net&output=plain
>>
>> To make dictionary attack even more difficult, you can use stretching
>> (aka iteration) - instead of just calculating the hash once, you iterate
>> it  $2^N$  times for some N. Iterate means you calculate the hash of the
>> password+salt, then the hash of that hash, then the hash of last hash
>> and so on. The point of the exercise is as follows: when you verify the
>> password, you need to perform this iteration only once. Suppose it takes
>> you a second to do that - not too terrible for the user to wait.
>> However, the attacker perapring the dictionary must do the iteration for
>> each password/salt combination, and those seconds start to add up. If
>> the unsalted password could be attacked in minutes, salted one in weeks,
>> then for salted and stretched it might take years or decades.
>> --
>> With best wishes,
>> Igor Tandetnik
>>
>> With sufficient thrust, pigs fly just fine. However, this is not
>> necessarily a good idea. It is hard to be sure where they are going to
>> land, and it could be dangerous sitting under them as they fly
>> overhead. -- RFC 1925
>>
>>
>>
```