

Re: How good an encryption algorithm is this?

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2004-11/0918.html>

From: Igor Tandetnik (itandetnik_at_mvps.org)

Date: 11/23/04

Date: Tue, 23 Nov 2004 14:12:52 -0500

"Bonj" <benjtaylor@hotpop.d0t.com> wrote in message news:OuHf31M0EHA.2716@TK2MSFTNGP14.phx.gbl

- > This is run twice, so I have two arrays of 256 bytes, say key1 and
- > key2. These are then hardcoded into the C++ encryption algorithm.
- > Then, the C++ encryption algorithm goes as such:
- > It memcpys the `_TCHAR` array to a byte array, then loops round each
- > byte of this array.
- > For each byte, it gets the value of `key1[n]` (where `n` is the byte
- > number), and calls this '`b_current_indir`' (the starting 'indirection
- > level'). Then, it gets the value of `key2[n]` and calls this 'levels' –
- > the
- > number of indirection levels.
- > Then, an inner loop runs 'levels' times – and on each loop the
- > following happens: the current byte of the data to be encrypted
- > (dictated by the outer loop) is XORed with `key2[b_current_indir]`, and
- > THEN, `b_current_indir` is reassigned to take on the value of
- > `key2[b_current_indir]`.

Note two facts. First, XOR is associative and commutative, that is, a sequence of XOR's can be executed in any order without changing the result. Second, $x \text{ XOR } x == 0$ for any x , and $x \text{ XOR } 0 == x$ for any x .

Suppose `key2[n]` happens to be even. Let the data byte at position n be x . Then your inner loop, the one that loops 'levels' times, XOR's x with itself an even number of times (the result is 0), then XORs some key material on top of that. The result does not depend on x at all. Which means, there's no way to recover plain text from ciphertext – the data is irretrievably lost in encryption.

Let's assume you fix it by simply XOR'ing x (the data byte) once. Now, the operations of the inner loop don't depend on x at all. You can represent an operation on byte x at position n as $(x \text{ XOR } K(n))$, where $K(n)$ depends only on n and key material (the bytes of `key1` and `key2`). You can as well precalculate $K(n)$ and do a simple XOR without any loop, and it won't change the result.

Thus, this modification of your algorithm is equivalent to a naive XOR

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

cipher – XORing the input data with a fixed key. If I happen to know both a plaintext and a ciphertext for some message (known–plaintext attack), I'll just XOR them together and get the key (note that the attacker often has access to at least some bytes of plaintext, since many messages contain known signatures in fixed places, like protocol headers and such). Even if I don't, a simple XOR cipher is a particular case of polyalphabetic aka Vigenere cipher – all the rage in 16–17th centuries until broken in 1863.

<http://www.trincoll.edu/depts/cpsc/cryptography/vigenere.html>

--

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