

Forum:[Hack Forums](#)

Thread: [DCryption | A File Encryption Algorithm | OTP Encryption Example](#)

-Max - February 22, 2013, 9:55 pm

I was playing around with some code and came up with this script. It randomly encrypts a file of your choice using a new random key each time.

There really isn't much else to it, you just have to see it for yourself!

Source: <https://github.com/Max00355/DCryption>

So, yeah, go ahead and tell me what's wrong with it :D

Oh, and yeah, I realize I didn't use OOP... there is nothing wrong with that. I wanted to try a different style from what I normally do.

L0iz - February 23, 2013, 2:33 am

Well that's one of the most inefficient one-time pad implementation I've seen.

Using smaller ints for random key and/or a modulo of 26 would yield the same level of security (unbreakable without the key, unbreakable as in truly unbreakable).

Then you can also change ints back to chars, to further decrease file size.

-Max - February 23, 2013, 7:19 am

(Feb 23, 2013 03:33 AM)L0iz Wrote: [\[CITE\]](#) Well that's one of the most inefficient one-time pad implementation I've seen.

Using smaller ints for random key and/or a modulo of 26 would yield the same level of security (unbreakable without the key, unbreakable as in truly unbreakable).

Then you can also change ints back to chars, to further decrease file size. Oh, I know, believe me! It was never meant for practical use.

Also, what's the name of this type of encryption? One-time pad you said? I didn't do any research I just thought of it.

EDIT: Wow, it already exists, interesting. http://en.wikipedia.org/wiki/One-time_pad That's almost exactly what I have here.

L0iz - February 23, 2013, 10:03 am

Yeah, it's exactly what you have there. :D

Quite interesting thingy, so simple and so secure.

-Max - February 23, 2013, 12:59 pm

(Feb 23, 2013 11:03 AM)L0iz Wrote: [\[CITE\]](#) Yeah, it's exactly what you have there. :D

Quite interesting thingy, so simple and so secure.

Yeah! I didn't realize! It's apparently impossible to crack. Interesting, very interesting.

Curator - February 24, 2013, 8:06 pm

(Feb 23, 2013 01:59 PM)Max00355 Wrote: [\[CITE\]](#) Yeah! I didn't realize! It's apparently impossible to crack. Interesting, very interesting. Impossible to crack if the storage and transmission of the key is perfectly secure, AND if the key is never re-used to encrypt a different message.

-Max - February 25, 2013, 11:26 am

(Feb 24, 2013 09:06 PM)Curator Wrote: [\[CITE\]](#) Impossible to crack if the storage and transmission of the key is perfectly secure, AND if the key is never re-used to encrypt a different message.

Well, yes, that is true. It's very interesting.

framk - February 27, 2013, 4:06 am

It's very nice as a little fun project, but it's horribly insecure. Sure, use it to hide files from co-workers or friends, but if you're going to get serious with your file incryption, nothing beats truecrypt.

Good job, OP.

L0iz - February 27, 2013, 6:09 am

(Feb 27, 2013 05:06 AM)framk Wrote: [\[CITE\]](#) It's very nice as a little fun project, but it's horribly insecure. Sure, use it to hide files from co-workers or friends, but if you're going to get serious with your file incryption, nothing beats truecrypt.

Good job, OP.

Actually it's quite the contrary. It's perfectly secure.

As already said the one time pad is unbreakable if the same key is not used twice.

On the other hand truecrypt mostly uses AES, which can be bruteforced. Not in a reasonable time (far from it), but still.

Which makes this infinite times more secure than truecrypt.

Of course it's practically much less useful, because of the key files, compared to a simple password for truecrypt.

Gpure - February 27, 2013, 11:01 am

Oh this is text encryption I thought you meant executable encryption. Quick question:

On the PHP Code

```
<?php
```

```
open (filearg, "wb/rb")
```

What does the wb and rb do? Is it permissions?

-Max - February 27, 2013, 12:14 pm

(Feb 27, 2013 12:01 PM)Gpure Wrote: [\[CITE\]](#) Oh this is text encryption I thought you meant executable encryption. Quick question:

On the PHP Code

```
<?php
```

```
open (filearg, "wb/rb")
```

What does the wb and rb do? Is it permissions?

It means open in binary mode. It's useful when opening files such as .exe files. Prevents them from corrupting. I usually open everything in binary mode, not sure I should, but I do.

Warren G - March 4, 2013, 7:05 pm

This looks awesome I can't wait to try it out.

K4Fr - July 24, 2013, 10:52 am

random.randint is bad idea for this use. For OTP to be secure it must be a RANDOM One-Time Pad. Random functions are not cryptographically secure, it's internal state is not considered secure (might be based on time solely) etc. Also it's pseudo-random, that is - numbers generated are based on initial 'seed' - so you in fact don't need a whole 'key' file, it would be sufficient to keep a small (64

bit? 32 bit?) seed. (<http://docs.python.org/dev/library/random.html> - "However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes") And I don't like the idea of adding 10000 to 0-255 ranged ASCII and then encoding it back as 0-255 ASCII. Get 8 bit OTP and XOR with letter. Still - treat as fun and you're safe. :P An acceptable (but not completely unbreakable) approach would be to get a seed from /dev/random (256 bits) and use it as a key for AES in CTR mode to generate pseudo random data but in cryptographically-secure manner. Also keep in mind that any approach working like that allows a potential 'attacker' to flip bits in decrypted plaintext at will. It might be usable for him. File would still need to be somehow authenticated. In your case you can in a controlled way increment/decrement letter in the alphabet. Change A to B etc. With xoring - you can flip bits. The normal bit-flipping attack for a stream cipher: http://en.wikipedia.org/wiki/Stream_cipher_attack your cipher is essentially a stream cipher. The key is the seed to the PRNG. (Your implementation is not a OTP - because you used random.randint()). I suggest you read about stream ciphers. The idea is similar to a OTP)

-Max - July 24, 2013, 12:05 pm

(Jul 24, 2013 10:52 AM)K4Fr Wrote: [\[CITE\]](#) random.randint is bad idea for this use. For OTP to be secure it must be a RANDOM One-Time Pad. Random functions are not cryptographically secure, it's internal state is not considered secure (might be based on time solely) etc. Also it's pseudo-random, that is - numbers generated are based on initial 'seed' - so you in fact don't need a whole 'key' file, it would be sufficient to keep a small (64 bit? 32 bit?) seed. (<http://docs.python.org/dev/library/random.html> - "However, being completely deterministic, it is not suitable for all purposes, and is completely unsuitable for cryptographic purposes") And I don't like the idea of adding 10000 to 0-255 ranged ASCII and then encoding it back as 0-255 ASCII. Get 8 bit OTP and XOR with letter. Still - treat as fun and you're safe. :P An acceptable (but not completely unbreakable) approach would be to get a seed from /dev/random (256 bits) and use it as a key for AES in CTR mode to generate pseudo random data but in cryptographically-secure manner. Also keep in mind that any approach working like that allows a potential 'attacker' to flip bits in decrypted plaintext at will. It might be usable for him. File would still need to be somehow authenticated. In your case you can in a controlled way increment/decrement letter in the alphabet. Change A to B etc. With xoring - you can flip bits. The normal bit-flipping attack for a stream cipher: http://en.wikipedia.org/wiki/Stream_cipher_attack your cipher is essentially a stream cipher. The key is the seed to the PRNG. (Your implementation is not a OTP - because you used random.randint()). I suggest you read about stream ciphers. The idea is similar to a OTP)

Although I appreciate the post, this thread is very old and it is stupid to dig it up from it's grave...