

Forum:[Hack Forums](#)

Thread: [How To Code Basic Cryptors III: The DES algorithm](#)

n3bul0n - April 11, 2013, 6:10 am

Hello HF,

the third part of my series on cryptography in C is all about DES. I even made a custom implementation for you, but this is where the fun stops. I coded 831 lines if I remember correctly, completely uncommented. The variable names are not funny anymore, because debugging that would be a pain in the ass. I will not post the code here, however, because I do not think it is that helpful. If you want it, however, just drop me a PM or post in this thread.

What I will do today is to explain the concept of DES and a basic overview of what you will need to code a Cryptor based on that algo yourself. If there are any mistakes, feel free to correct me. If anything is unclear, it might be my english or my surprising sobriety - but I tend to write very cryptic at times, too, I fear.

It is not that much C-related anymore, because I will not post snippets of code, but the algorithm is definitely codable relatively easy and I will look through all the codes you send me if you want me to check your implementations for flaws. As I said before, I also have a custom implementation for those too lazy to code.

Table Of Contents

Spoiler ([Click to View](#))

1. Beginning rant
2. Commented walkthrough through the algorithm
 - 2.1. The key
 - 2.2. Permutation - what is that
 - 2.3. The feistel-Function in detail
 - 2.4. The s-boxes - making 4 out of 6
 - 2.5. Another permutation - really?
3. Now that you're done understanding...
4. In the end
5. Literature used

Beginning rant

Firstly, a little checklist of what your coding abilities should include in order to code a working implementation of DES:

Spoiler ([Click to View](#))

- fluent knowledge of the C syntax
- fluent knowledge of loops, function calls and pointers
- knowledge of the mechanisms of two-dimensional arrays (g4143 will remember my ignorance towards that concept maybe)

- knowledge of the following libraries (those are the ones I used):

<stdio.h>

<conio.h>

<string.h>

<malloc.h>

<stdlib.h>

<math.h>

- a good grasp on mathematical concepts (what is XOR?)

- knowledge of the binary and hexadecimal system

- the absence of fear

DES was designed by IBM in the 1970s, if I recall correctly (I do, as [wikipedia](#) says). It was a great algorithm back then and is until now, even if there appeared some flaws as computer technology made its progress to GUIs and fancy-ass OSs and games.

I won't bother you with ye olden times, read it on wikipedia, if you're really interested. What I will teach you is the mechanisms and how the coder should see it.

IMPORTANT! DES is a symmetric algorithm, when you have the key, you are able to decypher it in no time. Please make sure if you are really going to use it (it's flawed! I mean it!), make sure to store the key somewhere save, preferably your brain. /IMPORTANT!

Walkthrough through the algorithm of DES

The algo as a picture:

Spoiler ([Click to View](#))

Figure 1

[img]http://s17.postimg.org/xqa0s2ogv/DES.png[/img]

Credits go to Andre Pawlowski and Wolfgang Ertel

In the beginning, there was a plain text. And the plain text had to be divided into blocks of 64 bit, because DES worked as a [block cipher](#). We ask for a key, then we go through the constant IP, the first permutation which is inverse to the FP at the end. That makes both of them quite useless, as they undo each other. But they were needed back in the 70s because of the computational structure (is this even a word?) back then.

We then split the block in halves, making it 32-bit blocks. We then prepare for 16 rounds of encrypting, following the following scheme(also to be seen in the picture): The right block runs through a function f (the Feistel function), then both of the halves are XOR'd and change their places (right to left, left to right). Then they go through that procedure another time, a total of 16 times. Lastly they are permuted with the constant FP and we are done. Now, the details:

The key

The key used has 64 bits. Firstly, we shorten it to 56 bits by leaving out every 8th bit (it's a parity bit, anyway). Now it is divided into two 28-bit blocks A0 and B0 (we know that already from above). In every round, the blocks are, depending on their round-index i, rotated left by one or two bits. This is specified in the standard. They go through a compression permutation which shortens the halves again, this time to 24 bits. This, again, works with a constant s-box(see next subchapter, 2.2.). We

now have 48 bits, that's the length we desire. For every round, we choose a new key, so that in the end we have used all 56 key bits several times.

Permutation - what is that?

Permutating means rearranging. For that, we need, fixed boxes, so called s-boxes(substitution boxes). You have several of them in the DES algorithm: IP, FP, s-boxes in function f, the P-box. You can get those in the internet, so you'll only have to implement them as two-dimensional arrays. What they do is this: They include certain numbers (in the case of IP and FP from 1 to 64, the others are explained in detail below), every number is only there once. Assuming the numbers go like this: 58, 50, 21...62. Then the 58th bit of the input will be the first bit of the output, the 50th bit of the input will be the second... and the 62th bit of the input will be the last and 64th of the output. Understandable? I think so.

The feistel-Function in detail

The feistel function is the first but not last hard-to-understand-are-you-kidding-me detail to DES. Wikipedia has a pretty good picture describing the function, I will explain it:

Spoiler ([Click to View](#))

Figure 2

Key and right block as input. So far, so easy. What is figure E. It is the expansion permutation. It is used to make the 32-bit block a 48-bit block by duplicating half of the bits after the following scheme: take bit 4 and 5 and duplicate them; duplicate 4 to 5 and 7 and 5 to 6 and 8. Then take the 8th and 9th bit and so on. Then we XOR key and block together. After that: the most important point of DES: the s-boxes.

The s-boxes(substitution boxes) - making 4 out of 6

The transformation achieved with the s-boxes makes 32 bit out of 48 again. But it doesn't just take all those bits out randomly, but by the means of those boxes. The 48 bit we have at the moment are split up into 8 six bit blocks, each connected to an s-box(they are, like the IP and FP, constant for all of the 16 rounds). But what are those s-boxes? They are two-dimensional arrays of numbers and look a little like this:

Spoiler ([Click to View](#))

It's the best picture I've found, it's in German, but it's not like you'd have to read much.

Figure 3

[img]http://ig.cs.tu-berlin.de/oldstatic/ap/rg/1998-04/glossar/s-terms/s-box-4.gif[/img]

One of those numbers from 0 to 15, seen as a binary number, will be our 4bit output. We choose one of the values by interpreting the first and last bit of our respective 6bit block as numbers; this will be the row. The second, third, fourth and fifth bith choose our column. And voilà, we have the output.

Another permutation? Really?

Yes. If you were attentive, you'd notice that there is a P-box below everything else in Figure 2. This is another fixed permutation. But if you got that to work, you're finally done.

Now that youre done understanding...

How about some coding? You don't need many things beside patience, a hand for debugging and a few preformed boxes. You can find them on several sites in the interwebs, eg [here](#). Make sure you'll check every step at least with a printf-command at the end, so you can see if everything went right. Other than that, I hope you'll do it yourself. There are plenty of good implementations there already, though, for example [here](#) and [here](#).

They are both not perfect, but oh well, who is?

In the end

Lastly, I'd like to thank HF for giving me a platform for my tutorial.

I hope you enjoy reading it as much as I did writing it and again, if you find anything fishy or not-quite-understandable, please leave a note.

Have a nice day,

n3bul0n

Literature used(everything there is german, sorry):

Dirk Fox- Data Encryption Standard (2000)

Sandra Thielert - Kryptographische Algorithmen (2007)

Andre Pawlowski - Kryptographische Verschlüsselung mithilfe des DES-Verfahrens und die Übersetzung eines Textes durch ein selbstgeschriebenes Delphi-Programm (2005)

Immortal - April 11, 2013, 11:03 am

Great tutorial I look to seeing you, in the c/c++ section from now on :D

also the picture was not working for me on google chrome, but when I right clicked on the image and then viewed the link it showed just fine now it show like normal.

Also First!

n3bul0n - April 11, 2013, 11:29 am

(Apr 11, 2013 11:03 AM)Imm?rtal Wrote: [\[CITE\]](#) Great tutorial I look to seeing you, in the c/c++ section from now on :D

also the picture was not working for me on google chrome, but when I right clicked on the image and then viewed the link it showed just fine now it show like normal.

Also First!

Thanks for you input! Appreciate it.

The picture should be working now...

?Dante - April 12, 2013, 1:53 am

Nice, simple, and quick reference on the topic. Good job OP!

-Dante

